Abstract

This research concentrates on the design and analysis of an algorithm referred to as Virtual Network Configuration which uses predicted future states of a system for faster network configuration and management. Virtual Network Configuration is applied to the configuration of a wireless mobile Asynchronous Transfer Mode network. Virtual Network Configuration is built on techniques from parallel discrete event simulation merged with constraints from real-time systems and applied to mobile ATM configuration and handoff.

Configuration in a mobile network is a dynamic and continuous process. Factors such as load, distance, capacity and topology are all constantly changing in a mobile environment. The Virtual Network Configuration algorithm anticipates configuration changes and speeds the reconfiguration process by pre-computing and caching results. Virtual Network Configuration propagates local prediction results throughout the Virtual Network Configuration enhanced system. The Global Positioning System is an enabling technology for the use of Virtual Network Configuration in mobile networks because it provides location information and accurate time for each node.

This research has resulted in well defined structures for the encapsulation of physical processes within Logical Processs and a generic library for enhancing a system with Virtual Network Configuration. Enhancing an existing system with Virtual Network Configuration is straight forward assuming the existing physical processes do not

have side effects. The benefit of prediction is gained at the cost of additional traffic and processing. This research includes an analysis of Virtual Network Configuration and suggestions for optimization of the Virtual Network Configuration algorithm and its parameters.

Contents

List of Tables

List of Figures

Chapter 1

Thesis Overview

1.1 What is Virtual Network Configuration?

This research concentrates on the design and analysis of an algorithm referred to as Virtual Network Configuration which predicts local future states of a system and efficiently propagates the effects of state changes through the system. The focus of this research is on the use of VNC for the configuration of a wireless mobile Asynchronous Transfer Mode [25] network. Results from the area of Parallel Discrete Event Simulation and real-time systems are synthesized and applied to mobile Asynchronous Transfer Mode configuration and handoff. This is accomplished by propagating the effects of predicted local changes through the network via a parallel simulation technique. As a local predicted event is propagated through the network, results are cached within each Logical Process which will be needed once the predicted change occurs.

Configuration in a mobile network is a dynamic and continuous process. Factors such as load, distance, capacity and topology are all constantly changing in a mobile environment. The Virtual Network Configuration algorithm anticipates configuration changes and speeds the reconfiguration process by pre-computing and caching results.

Configuration speedup occurs via three mechanisms: pre-computation and caching of events, concurrency, and the fact that it is possible for results to be generating faster than the critical path through the system. This is known as super-criticality and will be explained later. The Global Positioning System is an enabling technology for the implementation of this algorithm because it provides location information and accurate time for each node.

The Virtual Network Configuration algorithm can be explained by an example. A mobile node's direction, velocity, bandwidth, number of connections, past history and other factors can be used to approximate a new configuration sometime into the future. All actual configuration processes can begin to work ahead in time to pre-compute configuration values based on the position the remote node is expected to be some point in the future. If the prediction is incorrect, but within a given bound of the actual result, only some processing will have to be rolled back in time. For example, consider beamformed network links. Beamforming is a technique used to confine electro-magnetic radiation to a physically narrow path from the source to the destination node. Beamforming allows other links of the same frequency to be in operation at the same time and reduces the possibility of the link being detected by undesired listeners. In a beamforming network, the beamsteering process results may have to be adjusted, but the topology and many higher level requirements will remain correct. This working ahead and rolling back to adjust for error is a continuous process, depending on the tradeoff between allowable risk and distance into the future of predictions.

This research has resulted in well defined fields fields that are added to each existing message and additional structures that are added to existing processes. The benefit of prediction is gained at the cost of additional traffic and processing. An analysis of the efficiency and optimization of Virtual Network Configuration is presented in this thesis.

1.1.1 Thesis Outline

The research consists of three main tasks: algorithm design, analysis, and implementation and experimental validation. Optimization is an additional effort which permeates all of these tasks. The Virtual Network Configuration algorithm is fully developed, formally described, and simulated. The goal of the simulation is to determine the general characteristics of this algorithm and its suitability for use in network configuration and other applications. A method of assigning probabilities to predicted events is considered and optimizations to the basic algorithm are examined. In the analysis task, analytical results which describe the performance-cost tradeoff are determined.

Finally, in the third task of this research, the Rapidly Deployable Radio Network prototype is used to measure configuration and hand-off times without Virtual Network Configuration and compare these with the predictive Virtual Network Configuration method. The Network Control Protocol of Rapidly Deployable Radio Network [15] is enhanced so that the orderwire packet radios carry both real and virtual messages.

1.1.2 Algorithm Design

The effort to complete the algorithm design is pursued through two methods. A Maisie [6] simulation of Virtual Network Configuration has been developed to explore the effects of alternative methods of implementing the Virtual Network Configuration algorithm. A portable library written in C language which implements Virtual Network Configuration has been developed. The code was designed to allow any system to be easily enhanced with the Virtual Network Configuration algorithm. The method of prediction used in the driving process(es) is not of primary concern in this research. The amount of error, defined as the difference between the predicted and real values, is of concern, and this error probability is modeled. One of the first goals of the simulation is to examine the effects of overhead due to additional messages and rollback. The interaction of real and virtual message processing is of interest, particularly the method used for state adjustment in the Virtual Network Configuration protocol.

1.1.3 Virtual Network Configuration Analysis

Analytical results for the performance and costs of Virtual Network Configuration are determined in the analysis section. Analytical results are determined for the speedup gained by using Virtual Network Configuration as well as the overhead required. Speedup is the expected time predicted results, such as the Rapidly Deployable Radio Network beam table, are available with Virtual Network Configuration divided by the expected time required to obtain the results without Virtual Network Configuration.

Parameters such as lookahead (Λ) and tolerance (Θ) provide control over the acceptable risk and probability of error versus speed of configuration. Increasing Λ will better utilize the processors, but longer range predictions will be less accurate. Allowing more tolerance will reduce the probability of rollback and increase speed, but introduce a greater possibility of error. However, there exists control and flexibility with this algorithm. Thus a relationship between Θ, Λ, and costs including bandwidth are developed. Also, an attempt to develop a formal description or calculus for working with a Virtual Network Configuration system are undertaken using concepts from Petri Net Theory [92, 87].

1.1.4 Implementation and Experimental Validation

The goal of Virtual Network Configuration is to reduce the time required by each step of the configuration process. In order to verify the impact of Virtual Network Configuration, measurements are taken of configuration times for the Rapidly Deployable

Radio Network prototype. Configuration measurements include the time to establish the radio link (beamforming) up to and including network level configuration. The fact that each host in the Rapidly Deployable Radio Network network has its own Global Positioning System receiver greatly facilitates these measurements; mechanisms such as the Network Time Protocol [82] are not required. To perform these measurements, the Global Positioning System time is recorded on the source node before the message is sent, and again on the destination node immediately after the message is received.

Another step in configuration is determining the Edge Switch topology. Edge Switchs are intermediate nodes within the Rapidly Deployable Radio Network which have the capability of residing on the edge between fixed and wireless networks [17]. Edge Switchs serve as connections for Remote Nodes which are end nodes. Edge Switchs also have the capability of networking together to form a wireless network. Once the Edge Switch topology has been determined, Network Control Protocol Messages are exchanged which allow the beamformed radio link to be established. The protocol which uses the wireless virtual fiber is the Adaptive High Level Data Link Protocol Layer [53]. The Adaptive High Level Data Link Protocol Layer carries standard Asynchronous Transfer Mode cells; much of the remaining configuration occurs as in a wired network. In other words, above the Adaptive High Level Data Link Protocol level, the wireless network acts like a wired network, except for the significant impact of mobility.

Consideration has to be given to the costs incurred by Virtual Network Configuration in additional orderwire bandwidth consumed and additional CPU processing. Utilizing CPU processing for prediction when that CPU power would otherwise be wasted is a benefit, as long as processing does not become a bottleneck. However, the bandwidth used for configuration should be minimized.

A successful implementation shows a speedup in configuration and handoff while not exceeding the available orderwire bandwidth or causing a bottleneck in CPU processing. However, the impact of Virtual Network Configuration goes far beyond network configuration to any real time system which benefits from future state information. A subset of the Rapidly Deployable Radio Network architecture is enhanced with Virtual Network Configuration and a comparison made with non-Virtual Network Configuration configuration measurements. The Network Control Protocol, the configuration subsystem of Rapidly Deployable Radio Network which performs the configuration of the entire Rapidly Deployable Radio Network system, is enhanced with Virtual Network Configuration as a proto-type systems for this research.

1.1.5 Complexity in Self-Predictive Systems

The purpose of this section is to "step out of the box" and view from a few other perspectives the impossible fear of perfect prediction that we would like Virtual Network Configuration to perform. A perfectly tuned Virtual Network Configuration system would have a tolerance for error (Θ) of zero and $\forall\, t \;\, GVT(t) > t$, where $GVT(t)$ is the time into the future that the Virtual Network Configuration system has predicted. This means that the system should always be able to perfectly predict its future. However, as shown in Gödel's Theorem this cannot be accomplished. Thus, Virtual Network Configuration allows some prediction error to be present by introducing the notion of tolerance.

Another interesting, though troublesome concept which interferes with our knowledge of the future is Chaos. As an example, imagine Logical Processs which perform even simple computations such as the recursion in Equation 1.1, which is chaotic for certain values of r. Depending on the value of r and the initial value of X, there are periods of stability, a bifurcation, or jumping among stable values at $r = 3$, and completely unstable values in other iterations [51].

$$X_n = rX_{n-1}(1 - X_{n-1}) \tag{1.1}$$

Complexity theory deals with complicated systems with large numbers of processes (agents) whose global behaviors cannot be predicted simply from the rules of the agent interactions [24], [76]. Emergent systems lie at the edge of order and chaos and have resisted yielding to any known form of analysis. A simple, introductory chronicle of the history of the new field of emergent systems can be found in [118]. In [24], it is hypothesized that a true emergent system is one for which the optimal minimum means of prediction is simulation. But this author has not seen any mention of the effect of chaos and emergence on the operation of the simulation mechanism itself. An interesting tool for studying this may be the Swarm Simulation System [42] which is a simulation tool for studying systems with large numbers of agents interacting with each other within a dynamic environment. It has been used for studying emergence in economic, social, molecular, and ecological systems.

In [58], emergence in game theory is modeled and studied. Agents learn by receiving feedback from their environment, however, their environment consists of other agents who behave in exactly the same manner. The agents use relatively simple rules, but the behavior of the entire system is complex and still under study. This again points to the fact that a self predicting system composed of many Logical Processs may also have complex behavior.

In Virtual Network Configuration, the configuration system is operating with imperfect future position information. Such systems are analyzed in [47]. In [47], agents with imperfect information about their state can result in oscillatory and chaotic behavior for the entire system. Finally, [97] attempts to quantify when a complex system becomes chaotic, exhibiting self-similarity. This section has highlighted the many subtle complexities involved in attempting to build a self-predicting system.

Chapter 2

Wireless Mobile Network Challenges

2.1 Wireless ATM Architectures

Interest in developing wireless mobile telecommunication networks is increasing rapidly. There is expected to be a growing market for wireless mobile computing based on the fact that the mobile cellular voice communications market has grown rapidly in recent years and because of the ease and practicality offered by wireless mobile computing to the consumer. There were 43 million new wireless communications subscribers as of 1996. It is predicted that in 1997 there will be 60 million new subscribers which will surpass the number of new fixed network subscribers for the first time in history [30]. As more corporations begin to notice the potential profit in wireless mobile computing, they are attempting to obtain the results of the research done in this area. Besides the purely market driven requirements for mobile wireless computing, a wireless system can be rapidly deployed in a disaster situation bringing valuable information to rescue workers. A mobile wireless system can also be cheaply established in rugged terrain or in developing countries where the cost of laying cable would be prohibitive. Also, with a wireless mobile network, businesses will be able to more easily bring computing power to the location where they can gain the most profit. Finally, mobile wireless computing will be useful for the military.

This section provides background in order to understand the Virtual Network Configuration algorithm and the results of this research. This work is an application of a new algorithm to provide rapid wireless mobile Asynchronous Transfer Mode configuration. We will begin with an overview of wireless Asynchronous Transfer Mode architectures, because the manner in which the wireless links are implemented will impact the handoff performance of the predictive Virtual Network Configuration algorithm.

There appear to be two views on wireless Asynchronous Transfer Mode architecture. In one view, Asynchronous Transfer Mode cells should be maintained end-to-end including transmission over the wireless portion of the network. The other view is that Asynchronous Transfer Mode cells need not be preserved and a more efficient form of packetization should be used over the wireless network. The Asynchronous Transfer Mode cells are then reconstructed from the wireless packetization method after being received by the wireless destination.

Some thoughts on the latter view are provided in [80]. A Media Access Control protocol for wireless Asynchronous Transfer Mode is examined with a focus on Code Division Multiple Access access. One of the earliest proposals for a wireless Asynchronous Transfer Mode Architecture is described in [91]. This architecture appears to be the model on which the Asynchronous Transfer Mode Forum is basing its Wireless Asynchronous Transfer Mode Architecture [108]. It assumes a base station has an Asynchronous Transfer Mode Network Interface Unit (NIU) and a Personal Communications Network Network Interface Unit (PCN NIU). Various alternatives for a wireless Media Access Control are discussed and a Media Access Control frame is proposed which contains sequence numbers, service type, and a Cyclic Redundancy Checksum. Time of Expiry scheduling policy is proposed as a means for improving real-time data traffic handling. A related work which considers changes to Q.2931 [19] to support mobility is proposed in [88]. Q.2931 is the ITU standard for ATM signaling.

In the Rapidly Deployable Radio Network Project [17], Asynchronous Transfer Mode cells are transmitted over the wireless link intact within the Adaptive High Level Data Link Protocol protocol. A related paper analyzes fixed size Asynchronous Transfer Mode cells in a mobile environment [13]. In [13], a stochastic analysis of Asynchronous Transfer Mode buffer fill distribution is extended to a mobile environment. The performance of wireless networks with link level retransmissions is examined in [26]. These results indicate that application throughput improves with the presence of the link level retransmission protocol only when the wireless packet error rate exceeds a certain threshold.

In any wireless architecture, the occurrence of a hand-off must not interfere with Asynchronous Transfer Mode cell order. General handoff methods as well as some specific to Asynchronous Transfer Mode will be discussed.

2.2 Wireless Mobile Network Challenges

There is currently intense interest in adding seamless mobility to Asynchronous Transfer Mode networks; progress in that area is developing rapidly. However, there are several problems integrating seamless mobility with Asynchronous Transfer Mode which are difficult to solve. One of these problems is configuration in a rapidly changing network such as that presented by a mobile environment. This problem becomes especially acute in the inherently connection-oriented Asynchronous Transfer Mode environment. An end-to-end connection must be established before any transfer of data can take place and this setup time must be as short as possible yet allow for a constantly changing topology.

A large part of the difficulty in configuration is due to the difficulty caused by the handoff of a mobile node from one base station to another. In contrast to today's wireless mobile cellular voice networks, an efficient signaling mechanism for handoff in a seamless mobile Asynchronous Transfer Mode network has not yet been standardized and the effect of the handoff delay on Asynchronous Transfer Mode and higher layer protocols can be significant. This section provides an overview of some of the previous work related to the area of handoff for mobile Asynchronous Transfer Mode networks which tries to alleviate signaling traffic and maintain a seamless Asynchronous Transfer Mode connection. There are a variety of general methods for how and when handoff can occur. See Figure 2.1 for general examples of how handoff can occur. The Break-Make method of handoff breaks the mobile node's connection to the current base station before making a connection to the new base station. The Make-Break method of handoff is exactly the reverse of Break-Make. The Chaining method establishes a route through the previous base station then to the new base station after a handoff. Finally, either the Break-Make or Make-Break method can be used with the Chaining method resulting in a Combination method of handoff. An example of determining when handoff should be initiated is addressed in [96]. This algorithm is based on a least squares estimate of path loss parameters assuming location is known. It is shown that this technique reduces outages and performs better than an estimation windows method.

Handoff Type	**Description**
Break-Make	Break current connection first
Make-Break	Establish new connection first
Chaining	Route through old connection
Combination	Combine chaining with first two methods

Table 2.1: General Handoff Methods.

2.3 Approaches to Network Mobility

Mobile Networks are divided into two classes: connectionless and connection-oriented as shown in Figure 2.2. Connectionless networks pass datagrams without requiring any form of signaling to establish a channel. Connection-oriented networks require an initial signaling protocol to establish a communication channel.

In order to discuss the variety of disparate mechanisms proposed for mobile networking, a common terminology is required. Throughout this paper the acronym and terms defined in [41] are used; Mobile Terminal, Base Transceiver Station, Handoff, and Cell Site Switch are shown in Figure 2.1. The node where the old and new connections meet is called the Attach New to Old point. Bridging points are the nodes between which a connection change occurs. For example, the bridging nodes in a simple hand-off would be the Mobile Terminal and the Attach New to Old. Non-macro diversity is assumed throughout this paper, that is, only one transmission path per data stream exists from source to destination at any time.

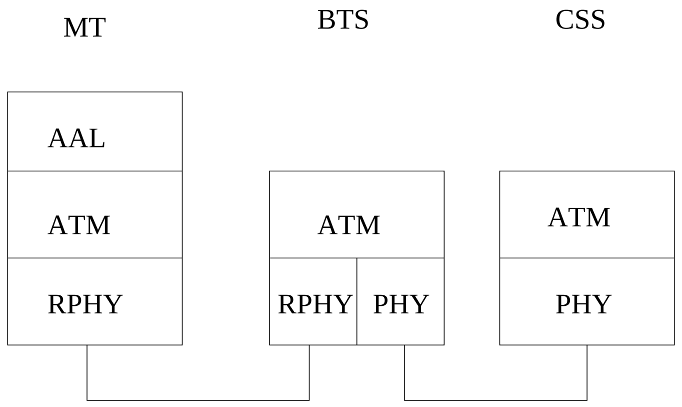


Figure 2.1: Basic Terminology.

Network Class	Sub Class	Sub Class	Description	Example
Connection-less			Handoffs require packet forwarding	
	Registration Agent		Packets forwarded by local agent	IP Mobility [86], CDPD [20]
Connection-Oriented			Handoffs require call-setup	
	Registration Agent		Calls forwarded by local agent	AMPS [28], GSM [31]
	Pivot Based		Distinct pivot BTS for handoff	
		Tree Based	Pre-established multicast connections	Acampora [1]
		Peer Based	Source and destination BTS negotiate handoff	ATM Forum [88]
	Adaptive Path		Maintains optimal routes during handoff	
		Predictive	Protocol stack and routing pre-configured by prediction	VNC [15] Full Mobility Architecture [71]
		Non-Predictive	Routes optimized after handoff occurs	BAHAMA [29]

Table 2.2: Overview of Mobile Communications Network Mechanisms.

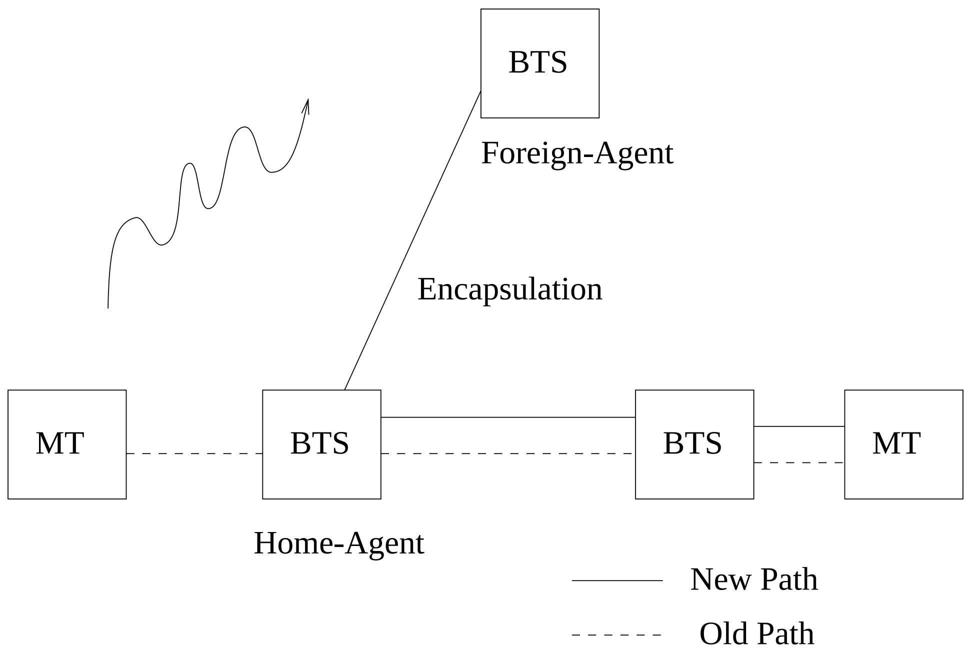


Figure 2.2: Registration Agent-Based Mobility Method.

2.3.1 Registration Agent-Based Mechanism

Figure 2.2 shows a hypothetical Registration Agent-Based mechanism similar to Mobile-IP which is described in this section. The initial bridge points are the Mobile Terminal (MT) and Base Transceiver Station (BTS) containing the Home Agent with the Home Agent Base Transceiver Station (BTS) serving as the Attach New to Old (ANO). Routing optimizations can remove the home Base Transceiver Station (BTS) from the path, with the source and destination BTSs as the bridge points. The Registration Agent-Based mechanisms contain a Home and Foreign Agent such as in Mobile-IP [86], explained next, or a Home Location Register (HLR) and Visiting Location Register (VLR) as in mobile cellular voice networks [28] and [31]. These agents maintain the information concerning whether a mobile node is currently associated with its home network or is a visiting mobile node, that is, not currently in its home network.

The IP Mobility Support Draft [86] is a good example of the connectionless registration mechanism. It describes enhancements that allow transparent routing of IP datagrams. This protocol uses an agent registration process to maintain home and current location information. All packets are routed to the home address as in the fixed network IP protocol. The home agent has a registration table which indicates whether the packet needs to be tunneled to a new address by encapsulating the packet within an IP packet bound for the new address, or to deliver the packet as in conventional fixed network IP. The mobile host registers with a foreign agent when it moves from its home network to a new network. The foreign agent receives the packet tunneled to it and delivers the packet to the mobile host by de-encapsulating the packet and forwarding the packet as in fixed network IP. IP Mobility Support allows for routers or entire networks to become mobile as a single unit. For example a ship or airplane could contain a network of mobile hosts and routers.

A Registration Agent-Based method for data over connection-oriented voice networks is the Cellular Digital Packet Data (CDPD) described in [20] and [61] protocol. The Cellular Digital Packet Data protocol operates much like the Mobile IP protocol for mobile cellular voice networks and is designed to co-exist with the Advanced Mobile Phone System (AMPS)[28]. Cellular Digital Packet Data uses registration and encapsulation to forward packets to the current location of the mobile host just as in Mobile-IP. Cellular Digital Packet Data was developed independently of Mobile-IP. Although the Cellular Digital Packet Data and Mobile-IP groups were aware of each others' existence, there has been surprisingly little interaction between the groups. This may be because the Cellular Digital Packet Data development group is a closed group focused on developing a standard to quickly meet commercial business requirements, whereas the Internet Engineering Task Force (IETF) group has moved more slowly in attempts to find better solutions to the technical and engineering problems involved.

An example of a connection-oriented mechanism for voice networks is Advanced

Mobile Phone System [28]. Advanced Mobile Phone System uses an Home Location Register (HLR) and Visiting Location Register (VLR) to store the mobiles' Mobile Identification Number (MIN) and Electronic Serial Number (ESN). The Home Location Register (HLR) and Visiting Location Register (VLR) reside on the Mobile Switching Center (MSC) which is connected to each mobile base station. A mobile node can determine whether it is in its home coverage area or in a new base station coverage area by comparing the base station identification, The Station Identification (SID), advertised by the base station, with a previously stored value. Registration and call routing are performed over Signaling System 7 (SS7). Call setup is routed first to the destination mobile's home Mobile Switching Center (MSC). The home Mobile Switching Center (MSC) will route the call to the Mobile Switching Center (MSC) which is currently serving the visiting mobile. If the visiting mobile node can be reached, the visiting mobile's Mobile Switching Center (MSC) will respond with a Temporary Directory Number (TDN). If the Mobile Switching Center (MSC) cannot reach the visiting mobile node, the Mobile Switching Center (MSC) will respond with a redirection indicating that the home Mobile Switching Center (MSC) must try another route or another number.

2.3.2 Tree-Based Mechanism

The Pivot-Based mobility mechanisms are characterized by a distinct node which is chosen to act as a pivot. A new route is chosen from the pivot node to the new base station after a handoff. Figure 2.3 shows the general Pivot-Based mechanism. Base Transceiver Station 1 is the Attach New to Old as the Mobile Terminal moves from Base Transceiver Station 2 to Base Transceiver Station 3. In the Tree-Based mechanisms, the Attach New to Old initiates and manages the hand-off. In the Peer-Based

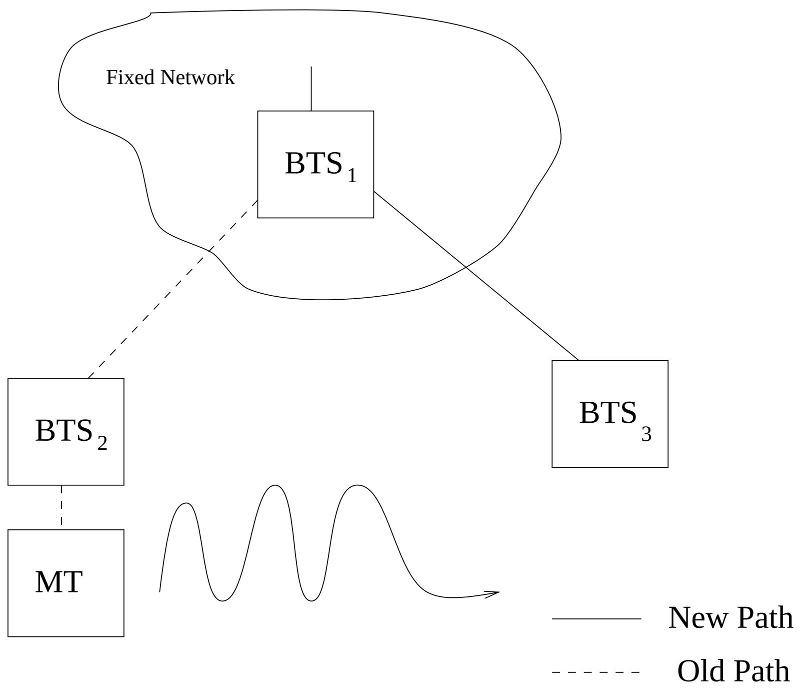


Figure 2.3: Pivot-Based Mobility Method.

mechanisms, Base Transceiver Station 2 and Base Transceiver Station 3 have an equal role in negotiating the hand-off.

The simplest of the Pivot-Based methods is the tree-based method which requires preallocation of resources but does not require Virtual Circuit setup and is thus efficient for performing hand-offs. A good description of a tree-based method is described in [1]. In this method, the tree is formed by Virtual Paths from a root Asynchronous Transfer Mode switch in the fixed network. The leaves of the tree are the base stations. The Asynchronous Transfer Mode switches within the tree are enhanced with translators which monitor incoming cells and change routes based on incoming cells. A handoff is implied by cells arriving from a given Virtual Circuit and port; this causes the switch to route cells along the new Virtual Circuit and port without any additional signaling. This method has been carried into more detail, such as considering Quality of Service, by [83].

The mobile Asynchronous Transfer Mode architecture described in [1] is based on a *virtual connection tree*. The root of the virtual connection tree is an Asynchronous Transfer Mode switch in the wired network whose leaves are the base stations. A mobile node will have a route first through the virtual connection tree root and then to either the wired network or back through the root to another mobile node. There can be more than one virtual connection tree throughout the network. When a mobile node joins a particular virtual connection tree, virtual circuits are established for that mobile node along every branch of the tree. These Virtual Circuits remain established while the mobile node resides within the virtual connection tree. Thus when the mobile node performs a handoff from one base station to another within the same virtual connection tree signaling is not required to establish a new Virtual Circuit. This works for Asynchronous Transfer Mode cells traveling in the forward direction relative to the mobile node, but cells in the reverse direction will still arrive at the mobile node's previous destination. This problem is solved by enhancing the virtual connection tree root Asynchronous Transfer Mode switch with a Virtual Circuit translation device. When the first Asynchronous Transfer Mode cell arrives at the root with a new virtual circuit identifier, the root Asynchronous Transfer Mode switch will update the routing table to forward cells in the reverse direction properly.

The virtual connection tree method has the advantage of not requiring any changes to the Asynchronous Transfer Mode cell structure or its interpretation. However, this method does have problems. Since a virtual connection tree maintains Virtual Circuits for all mobile nodes to every branch of the tree, there are many Virtual Circuits open which are not being used. In addition this leaves all potential Virtual Circuits open on all base stations. If many mobile nodes went to a single base station at the same time, they would all be immediately accepted potentially causing the base station to be overloaded.

2.3.3 Peer-Based Mechanism

The Peer-Based method is included in the Pivot-Based mobility class illustrated in Figure 2.3. The Peer-Based method assumes a fixed node which acts as a pivot, but it differs from the Tree-Based methods in that the base station nodes to which the mobile unit is associated before and after the hand-off may negotiate among themselves as peers in order to attempt to establish the same Quality of Service for the mobile unit.

Indirect-TCP (I-TCP) as described in [8] is an example of this in the connection-oriented TCP environment. I-TCP requires Mobile Support Router which have the ability to transfer images of sockets involved in an established TCP connection from one MSR to another. This transfer provides a seamless TCP connection as the mobile host performs a hand-off. The transfer is transparent to both the mobile node and a fixed network node. Clearly, one would expect a significant amount of overhead in copying the connected sockets from one Mobile Support Router to another. In addition, IP packets are lost during the hand-off due to the routing update which is required after the socket transfer. The routing update is required because the host at the opposite end of the connection which did not hand-off Fixed Host is connected to a new Mobile Support Router. Thus the FH and the new Mobile Support Router are the bridge nodes in this routing update. However, the results reported in [8] indicate that I-TCP performed better than regular TCP over Mobile-IP. This is explained as being primarily due to the TCP restart delay incurred by regular TCP.

Another Peer-Based mobile Asynchronous Transfer Mode mechanism which is analogous to the I-TCP method is the architecture for mobility support in wireless Asynchronous Transfer Mode Networks described in [88]. In [88] additions to Q.2931 are proposed which allow the Asynchronous Transfer Mode switch before, and the new Asynchronous Transfer Mode switch after a hand-off to negotiate a comparable Quality of Service for the mobile host and transfer state information in order to maintain cell order. The mechanism described in [88] is currently a strong candidate for standardization by the Asynchronous Transfer Mode Forum. If the Quality of Service can be maintained through the destination Asynchronous Transfer Mode switch, then the hand-off is seamless[1].

2.3.4 Adaptive Path-Based Mechanism

The Adaptive Path-based mechanisms do not require resource preallocation, but require more complex modifications of the signaling protocol. The routing is an integral part of the mechanism and maintains an optimal route between connections. There is no home or foreign agent; a distributed topology and routing protocol keep cell transfer tables accurate and consistent as topology changes.

An Adaptive Path-Based method of hand-off for Asynchronous Transfer Mode is described in [29, 56, 116]. This system consists of Portable Base Station and mobile users. Portable Base Stations are base stations which perform Asynchronous Transfer Mode cell **forwarding**, not Virtual Circuit Identifier/Virtual Path Identifier translation. This is a very important point, because it means that Asynchronous Transfer Mode cell forwarding, based on Virtual Path Identifier, occurs much like IP packet forwarding in Mobile-IP. This method tries to blend the best of the connectionless mechanisms into the connection-oriented environment. Portable Base Stations are connected via Virtual Path Trees which are preconfigured Asynchronous Transfer Mode Virtual Path. These trees can change based on the topology as described in the *Virtual Trees Routing Protocol* [57]. Since Virtual Path Trees are pre-established, no Virtual Path setup is required;

[1]Contribution [88] only mentions that data link information is transferred between Asynchronous Transfer Mode switches to maintain state information and that this guarantees in-sequence cell delivery. This would appear to require more explanation from the authors.

individual connections are established by simply choosing available Virtual Circuit on the source and destination nodes. The fact that Virtual Paths are maintained based on topology, not call-setup, allows fast hand-off since connection routes do not need to be established with each call-setup. The Virtual Path Identifier denotes a Portable Base Station address which is the root of the Virtual Path Identifier tree and a Portable Base Station can belong to more than one tree. A Homing Algorithm maintains cell order without centralized control or re-sequencing at any point in the network. The Homing Algorithm routes cells through the destination node's home Portable Base Station regardless of where the mobile node is currently located. The route will update to a more optimal route when it is possible to do so without causing cell reordering. This is done for Asynchronous Transfer Mode cells in both directions on a channel. This algorithm is considered an adaptive path algorithm because both the Virtual Path Trees and the mobile node's home Portable Base Station update gradually to result in more efficient routes.

2.3.5 Predictive-Based Mechanisms

Clearly, some type of prediction can be used in any mobile network mechanism. However, the Predictive-Based mechanism as used in this work refers to a mobile algorithm which makes use of predicted mobile location to pre-establish a significant portion of the post-handoff connection. These mechanisms are located under the Adaptive Path-Based mechanisms because Predictive Path-Based mechanisms can attempt to optimize the route based on predicted topology.

Many recently proposed mobile networking architectures and protocols involve predictive mobility management schemes. An optimization to a Mobile IP-like protocol using IP-Multicast is described in [99]. Hand-offs are anticipated and data is multicast to nodes within the neighborhood of the predicted handoff. These nodes intelligently buffer the data so that no matter where the mobile host (MH) re-associates after a handoff, no data will be lost. Another example [70] and [71] proposes deploying mobile floating agents which decouple services and resources from the underlying network. These agents would be pre-assigned and pre-connected to predicted user locations. Finally, [15] discusses a Predictive-Based Mobile Private Network-Network Interface. This mechanism inserts a lookahead capability into the topology portion of fixed network Private Network-Network Interface while maintaining as much of the fixed network Private Network-Network Interface standard as possible.

2.4 Comparison of Mechanisms

The applicability of these seamless mobile handoff mechanisms to Asynchronous Transfer Mode varies in the efficiency and the ability of each mechanism to maintain the Asynchronous Transfer Mode protocol standard end-to-end. A Registration Agent-Based mechanism for Asynchronous Transfer Mode at the cell level would require significant changes and overhead to the standard Asynchronous Transfer Mode protocol in order to implement tunneling, however, the Adaptive Path-Based mechanism, [29], has found an interesting middle ground discussed later in this section. The Tree-Based mechanism is efficient but it is static; routes do not adjust within the network to form the most efficient paths, base stations are assumed to be stationary, and all channels must be preconfigured for each mobile at each base station. Each base station will be required to allocate many more VCs than are really necessary. An advantage of the tree-based method is that it maintains the Asynchronous Transfer Mode protocol end-to-end. The Adaptive Path-Based mechanism described previously, [29], is more flexible than the Tree-Based method. It maintains Virtual Path Trees which remain consistent with the changing topology. However it requires modifications to the protocol such as using the Generic Flow Control to contain a sequence number and an end-of-transmission indicator within the packet. The Homing Algorithm used by this adaptive path-based algorithm allows the routing to be adaptive to form efficient paths; however, the paths adapt relatively slowly. The reason for the slow adaptation is that all cells sent before a handoff must be routed through the old Portable Base Station in order insure cell order. Finally, the mechanism in [29] requires cell forwarding rather than switching and a set of signaling and control algorithms which are very different from the standards currently in use. The Connection-Oriented Peer-Based method of mobile Asynchronous Transfer Mode handoff described in this paper, [88], although more flexible than the Tree-Based method, leaves some open questions regarding the overhead of the peer Asynchronous Transfer Mode switch negotiation and the maintenance of Asynchronous Transfer Mode cell order. The Adaptive Path-Based method in [29] integrates topology and routing in a more fundamental manner and would appear to handle the long term operation of the wireless network more efficiently. Also, the enhancement to the Q.2931 signaling protocol to implement the Peer-Based handoff scheme does not handle route optimization. Route optimization may be required as a final step to this method in order to make it comparable with [29].

The metrics of interest which are relatively easy to measure include the handoff delay and the additional network load due to signaling. Additional metrics which are less easy to quantify are the flexibility of the method in handling all cases of mobility and how closely the method aligns with existing standards. The flexibility of the method involves such criteria as whether the method can handle every hand-off situation and whether the method requires a fixed network connection or whether it can support a completely wireless environment. Much more experimental validation and measurement is required of these methods.

The contribution of this section has been to categorize mobile wireless Asynchronous Transfer Mode configuration as well as to compare and contrast them. There is a trade-off between flexibility, handoff delay, and maintenance of the Asynchronous Transfer Mode protocol standard, and amount of increase in signal load. The proposals range from the Tree-Based method which is fast but not very flexible to the Adaptive Path and Peer-Based mechanisms which are more flexible, but require more overhead for each handoff.

Putting aside the requirement for adhering to the Asynchronous Transfer Mode protocol standard, the best candidate at this time appears to be the Adaptive Path-Based mechanism described in [29]. Although the Predictive-Based mechanisms may prove better, the Predictive-Based mechanisms are behind [29] in actual development and testing. This work on Virtual Network Configuration, a Predictive-Based mechanism, will help to remedy that situation. The Adaptive Path-Based mechanism by definition treats topology and routing in a fundamental and efficient manner, not as additional steps separate from the mobile algorithm. Finally, [29], applies techniques, namely cell forwarding, from the already proven connectionless approach of Mobile-IP to the connection-oriented environment of mobile Asynchronous Transfer Mode.

In Rapidly Deployable Radio Network, the orderwire determines and controls handoff. The Network Control Protocol controls the operation of the orderwire and is discussed in more detail in the next section.

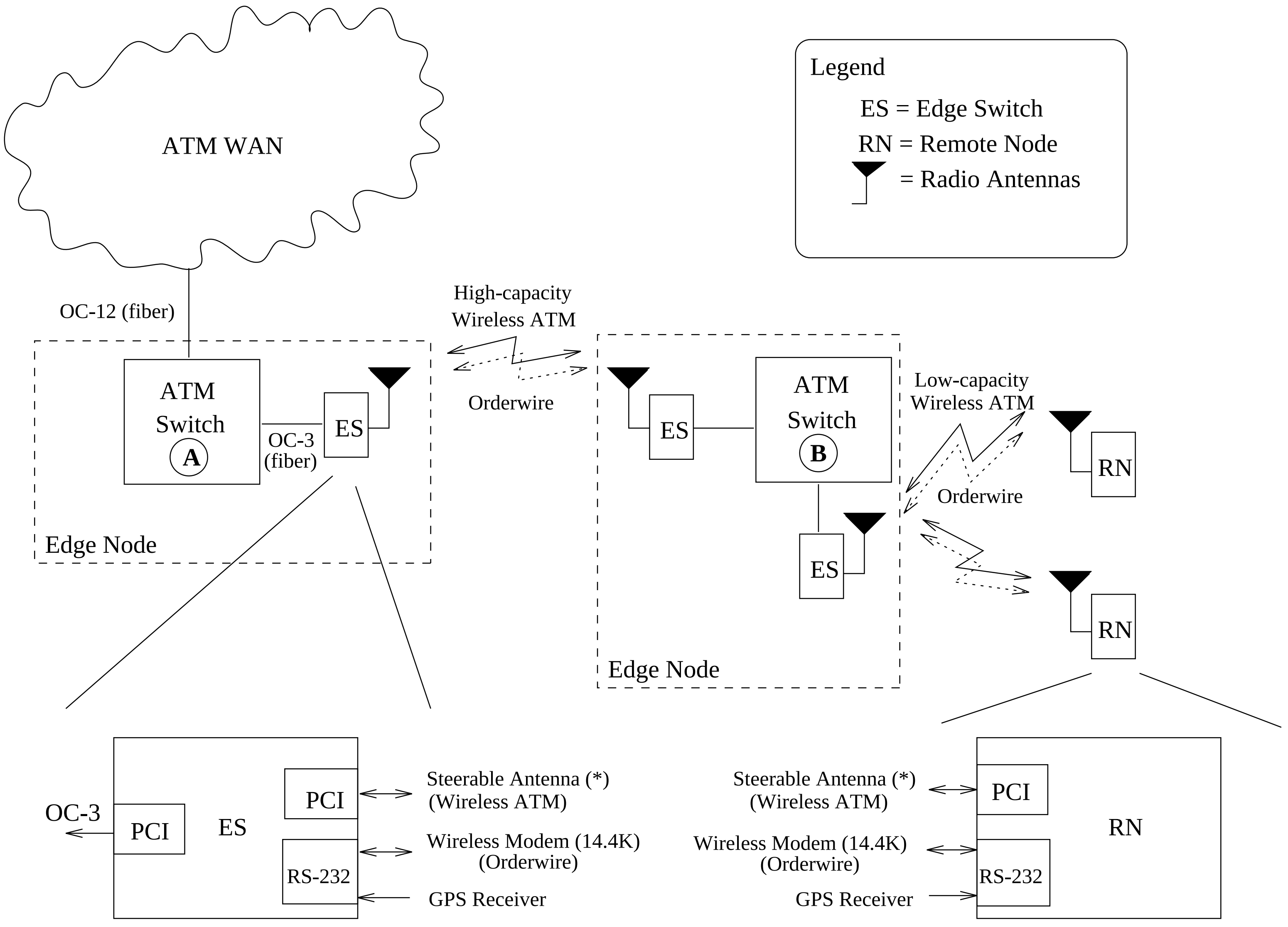
2.5 Overview of the Rapidly Deployable Radio Network System and Orderwire

It is important to understand the Rapidly Deployable Radio Network system which the orderwire serves and the details of the orderwire operation. A brief review of the Rapidly Deployable Radio Network system begins with one of the earliest proposals for a wireless Asynchronous Transfer Mode architecture which is described in [91]. In [91], various alternatives for a wireless Media Access Control are discussed and a Media Access Control frame is proposed. The Media Access Control contains sequence numbers, service type, and a Time of Expiry scheduling policy as a means for improving real-time data traffic handling. A related work which considers changes to Q.2931 [19] to support mobility is proposed in [88]. A Media Access Control protocol for wireless Asynchronous Transfer Mode is examined in [80] with a focus on Code Division Multiple Access in which Asynchronous Transfer Mode cells are not preserved allowing a more efficient form of packetization over the wireless network links. The Asynchronous Transfer Mode cells are reconstructed from the wireless packetization method after being received by the destination. The Rapidly Deployable Radio Network architecture maintains standard Asynchronous Transfer Mode cells through the wireless links. Research work on wireless Asynchronous Transfer Mode LANs have been described in [23] and [3]. The mobile wireless Asynchronous Transfer Mode Rapidly Deployable Radio Network differs from these LANs because the Rapidly Deployable Radio Network uses point-to-point radio communication over much longer distances. The system described in [29] and [56] consists of Portable Base Station and mobile users. PBSs are base stations which perform Asynchronous Transfer Mode cell switching and are connected via Virtual Path Trees which are preconfigured Asynchronous Transfer Mode Virtual Path. These trees can change based on the topology as described in the *Virtual Trees Routing Protocol* [57]. However, Asynchronous Transfer Mode cells are forwarded along the Virtual Path Tree rather than switched, which differs from the Asynchronous Transfer Mode standard. An alternative mobile wireless Asynchronous Transfer Mode system is presented in this paper which consists of a mobile PNNI architecture based on a general purpose predictive mechanism known as Virtual Network Configuration that allows seamless rapid handoff.

The objective of the Rapidly Deployable Radio Network effort is to create an Asynchronous Transfer Mode-based wireless communication system that will be adaptive at both the link and network levels to allow for rapid deployment and response to a changing environment. The objective of the architecture is to use an adaptive point-to-point topology to gain the advantages of Asynchronous Transfer Mode for wireless networks. A prototype of this system has been implemented and will be demonstrated over a wide area network. The system adapts to its environment and can automatically arrange itself into a high capacity, fault tolerant, and reliable network. The Rapidly Deployable Radio Network architecture is composed of two overlayed networks:

- A low bandwidth, low power omni-directional network for location dissemination, switch coordination, and management which is the orderwire network described in this section,
- A "cellular-like" system for multiple end-user access to the switch using directional antennas for spatial reuse, and and a high capacity, highly directional, multiple beam network for switch-to-switch communication.

The network currently consists of two types of nodes, Edge Nodes and Remote Nodes as shown in Figure 2.4. Edge Nodes were designed to reside on the edge of a wired network and provide access to the wireless network; however, Edge Nodes also have wireless links. The Edge Node components include Edge Switches and optionally



(*) Low Speed: 2Mbps, High Speed (planned): 10Mbps

Figure 2.4: RDRN High-level Architecture.

an Asynchronous Transfer Mode switch, a radio handling the Asynchronous Transfer Mode-based communications, a packet radio for the low speed orderwire running a protocol based on X.25 (AX.25), Global Positioning System receiver, and a processor. Remote nodes (Remote Node) consist of the above, but do not contain an Asynchronous Transfer Mode switch. The Edge Nodes and Remote Nodes also include a phased array steerable antenna. The Rapidly Deployable Radio Network uses position information from the Global Positioning System for steering antenna beams toward nearby nodes and nulls toward interferers, thus establishing the high capacity links as illustrated in Figure 2.5. Figure 2.5 highlights an Edge Switch (center of figure) with its omni-directional transmit and receive orderwire antenna and an omni-directional receive and directional transmit Asynchronous Transfer Mode-based links. Note that two Remote Nodes share the same 45^o beam from the Edge Switch and that four distinct frequencies are in use to avoid interference. The decision involving which beams to establish and which frequencies to use is made by the topology algorithm which is discussed in a later section.

The Edge Switch has the capability of switching Asynchronous Transfer Mode cells among connected Remote Nodes or passing the cells on to an Asynchronous Transfer Mode switch to wire-based nodes. Note that the differences between an Edge Switch and Remote Node are that the Edge Switch performs switching and has the capability of higher speed radio links with other Edge Switches as well as connections to wired Asynchronous Transfer Mode networks.

The orderwire network uses a low power, omni-directional channel, operating at 19200 bps, for signaling, control, and communicating node locations to other network elements. The orderwire aids link establishment between the Edge Switches and between the Remote Nodes and Edge Switches, tracking remote nodes and determining link quality. The orderwire operates over packet radios and is part of the Network Con-

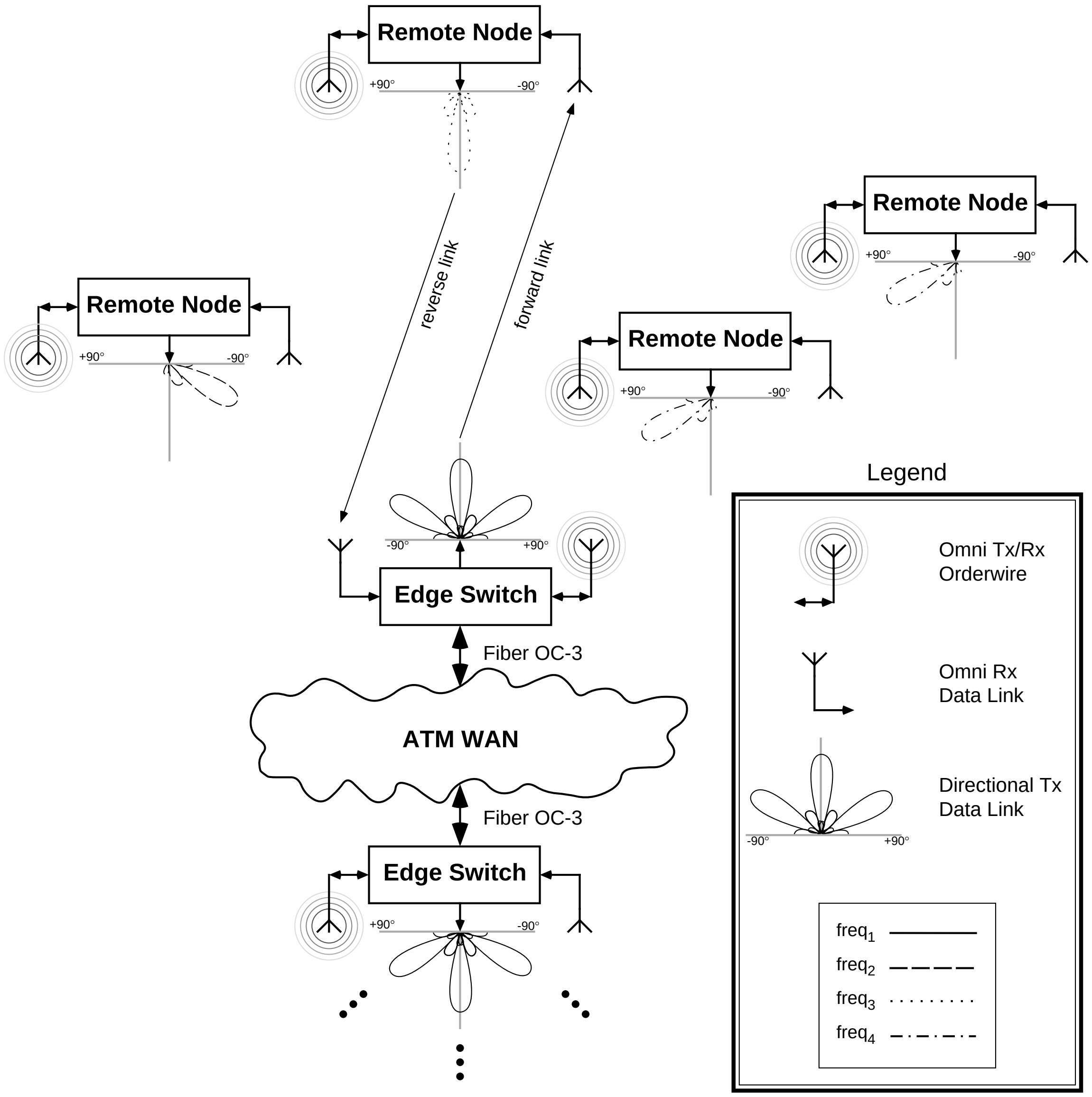


Figure 2.5: Rapidly Deployable Radio Network Component Overview.

trol Protocol[2]. An example of the user data and orderwire network topology is shown in Figure 2.6. In this figure, an Edge Switch serves as a link between a wired and wireless network, while the remaining Edge Switches act as wireless switches. The protocol stack for this network is shown in Figure 2.7.

The focus of this section is on the Network Control Protocol and in particular on the orderwire network and protocols used to configure the network. This includes protocol layer configuration, link quality, hand-off, and host/switch assignment along with information provided by the Global Positioning System system such as position and time. The details of the user data network will be covered in this paper only in terms of services required from, and interactions with, the Network Control Protocol.

2.5.1 Network Control Protocol

At the physical level the orderwire is used to exchange position, time and link quality information and to establish the wireless Asynchronous Transfer Mode connections. The process of setting up the wireless connections involves setting up links between Edge Switches and between Edge Switches and Remote Nodes. Error conditions are discussed at the end of this Section and in Section 2.5.2.

The network has one Master Edge Switch, which runs the topology configuration algorithm [39] and distributes the resulting topology information to all the connected Edge Switches over connection-oriented orderwire packet radio links. In the current prototype the connection-oriented link layer for the orderwire uses AX.25 [48]. The term “AX.25 connection-oriented” in the following discussion means that the AX.25 Numbered Information Frames are used. Broadcast mode in AX.25 uses Unnumbered

[2]The Simple Network Management Protocol Management Information Base for the Network Control Protocol operation as well as live data from the running prototype Rapidly Deployable Radio Network system can be retrieved from http://www.ittc.ukans.edu/∼sbush/rdrn/ncp.html.

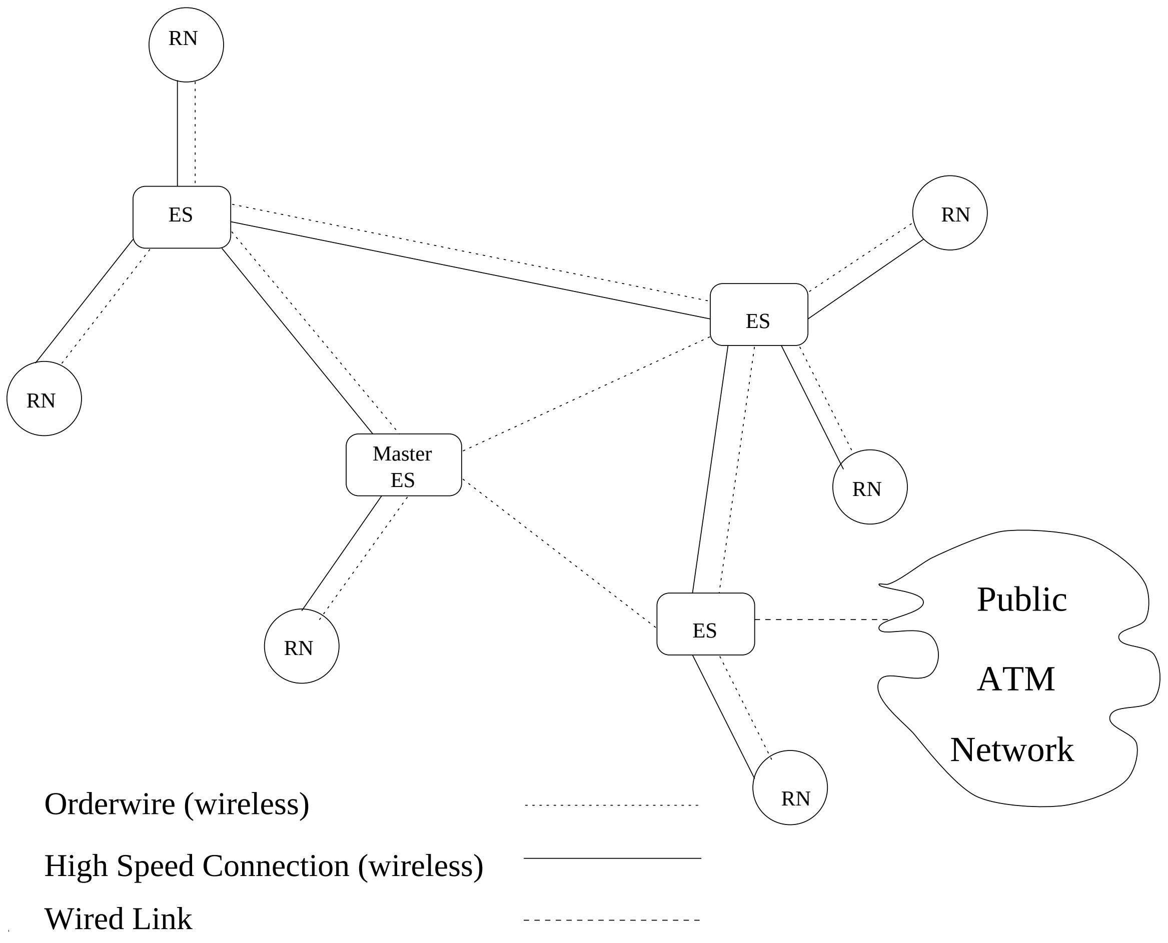
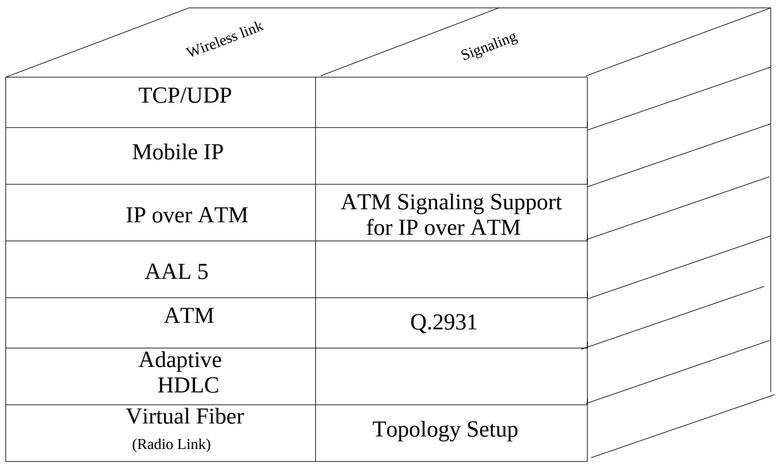


Figure 2.6: Example Orderwire Topology.



Wireless link	Signaling
TCP/UDP	
Mobile IP	
IP over ATM	ATM Signaling Support for IP over ATM
AAL 5	
ATM	Q.2931
Adaptive HDLC	
Virtual Fiber (Radio Link)	Topology Setup

Figure 2.7: Wireless Asynchronous Transfer Mode Protocol Stack.

Information Frames. The Master Edge Switch is the first active Edge Switch, and any Edge Switch has the capability of becoming a Master Edge Switch.

The following description of the Rapidly Deployable Radio Network Orderwire operation references the state transitions labeled in Table 2.3. The first Edge Switch to become active broadcasts its callsign and start-up-time in a **MYCALL** packet and listens for responses from any other Edge Switches (Label I1 in Table 2.3). In this prototype system, the packet radio callsign is assigned by the FCC and identifies the radio operator. Since it is assumed that this is the first active Edge Switch, there is no response within the given time period, T. At the end of T seconds, the Edge Switch rebroadcasts its **MYCALL** packet and waits another T seconds. At the end of $2T$ seconds, if there are still no responses from other Edge Switches, the Edge Switch activates and takes on the role of the Master Edge Switch (Label I4 in Table 2.3). If the first two or more Edge Switches start up within T seconds of each other, at the end of the interval T, the Edge Switches compare the start-up times in all the received **MYCALL** packets and the Edge Switch with the oldest start-up time becomes the Master Edge Switch. If the oldest two start-up times are equal, the Edge Switch with the lowest lexicographic callsign becomes the Master Edge Switch. In this system, accurate time stamps are provided by the Global Positioning System receiver.

Each successive Edge Switch that becomes active broadcasts its callsign in a **MYCALL** packet (Label I1 in Table 2.3). Upon receipt of a **MYCALL** packet, the Master Edge Switch extracts the callsign of the source, establishes an AX.25 connection-oriented link to the new Edge Switch and sends it a **NEWSWITCH** packet (Label I2 in Table 2.3). On receipt of the **NEWSWITCH** packet over the AX.25 connection-oriented orderwire link, the Edge Switch obtains its position from its Global Positioning System receiver and sends its position to the Master Edge Switch as a **SWITCHPOS** packet over the AX.25 connection-oriented orderwire link (Label I3 in Table 2.3). On receipt of a **SWITCHPOS** packet, the Master Edge Switch records the position of the new Edge Switch in its switch position table and runs the topology configuration algorithm [39] to determine the best possible interconnection of all the Edge Switches. The master then distributes the resulting information to all the Edge Switches in the form of a **TOPOLOGY** packet over the AX.25 connection-oriented orderwire links (Label I4 in Table 2.3). Each Edge Switch then uses this information to setup the Edge Switch links as specified by the topology algorithm. The Master Edge Switch also distributes a copy of its switch position table to all the Edge Switches over the AX.25 connection-oriented orderwire links which can be used to configure Remote Nodes as discussed below. The Edge Switch can then use the callsign information in the switch position table to setup any additional AX.25 connection-oriented orderwire packet radio links corresponding to the Edge Switch links required to exchange any link quality information. Thus, this scheme results in an AX.25 connection-oriented star network of orderwire links with the Master Edge Switch at the center of the star and the AX.25 connection-oriented orderwire links between those Edge Switches which have corresponding Edge Switch links as shown in Figure 2.6. The underlying AX.25 protocol generates error messages when a frame has failed successful transmission after a specified number of attempts. In the event of failure of the master node, which can be detected by listening for the AX-25 messages generated on node failure, the remaining Edge Switches exchange **MYCALL** packets (Label I1-I3 in Table 2.3), elect a new master node, and the network of Edge Switches is reconfigured using the topology configuration algorithm [39] (I4).

The Edge Switch/Remote Node operation is described by Tables 2.3 and 2.4. The packet contents are shown in Table 2.5. Each Remote Node that becomes active obtains its position from its Global Positioning System receiver and broadcasts its position as a **USER_POS** packet over the orderwire network (Label A1 in Table 2.4). The timeout value in Table 2.4 is the **USER_POS** update period. The **USER_POS** packet is received by all Edge Switches within range (Label A2 in Table 2.4). Each Edge Switch then computes the optimal grouping of Remote Nodes within a beam via an algorithm described in [40]. The beamforming algorithm returns the steering angles for each of the beams originating from the Edge Switch so that all the Remote Nodes are covered. If the Network Control Protocol determines that a beam and Time Division Multiple Access time slot are available to support the new Remote Node, the Edge Switch steers its beams so that all its connected Remote Nodes and the new Remote Node are covered (Label R1 in Table 2.4). The Edge Switch also records the new Remote Node's position in its user position table, establishes a AX.25 connection-oriented orderwire link to the new Remote Node, and sends the new Remote Node a **HANDOFF** packet with link setup information indicating that the Remote Node is associated with it (Label R2 in Table 2.4). If the new Remote Node cannot be accommodated, the Edge Switch sends it a **HANDOFF** packet with the callsign of the next closest Edge Switch, to which the Remote Node sends another **USER_POS** packet over a AX.25 connection-oriented orderwire link. This Edge Switch then uses the beamform algorithm to determine if it can handle the Remote Node.

This scheme uses feedback from the beamforming algorithm together with the distance information to configure the Remote Node. It should be noted that the underlying AX.25 protocol [48] provides error free transmissions over AX.25 connection-oriented orderwire links. Also the AX.25 connection-oriented orderwire link can be established from either end and the handshake mechanism for setting up such a link is handled by AX.25. If the Remote Node does not receive a **HANDOFF** packet within a given time it uses a retry mechanism to ensure successful broadcast of its **USER_POS** packet.

An AX.25 connection-oriented orderwire link is retained as long as a Remote Node is connected to a particular Edge Switch and a corresponding high-speed link exists between them to enable exchange of link quality information. The link can be removed when the mobile Remote Node migrates to another Edge Switch in case of a hand-off. Thus at the end of this network configuration process, three overlayed networks are established; an orderwire network, an Remote Node to Edge Switch network, and an Edge Switch to Edge Switch network. The orderwire network has links between the Master Edge Switch and every other active Edge Switch in a star configuration, links between Edge Switches connected by Edge Switch links as well as links between Remote Nodes and the Edge Switches to which they are connected, as shown in Figure 2.6. Raw pipes for the user data links between Remote Nodes and appropriate Edge Switches as well as for the user data links between Edge Switches are also established. Any Edge Switch which powers-up, powers-down, changes position, or becomes unusable due to any malfunction causes the entire Network Control Protocol system to reconfigure. Multiple Network Control Protocol groups will form when a set of nodes are out of range of each other. Edge Switch nodes which receive **NEWSWITCH** packets from multiple Master Edge Switches will join the group whose Master Edge Switch has the oldest start time or which has the lowest lexicographic callsign in case of equal start times. However, if digipeating is enabled, any Edge Switch node can act as a forwarder of messages for nodes which are out of range, which would enable a single group to be formed.

2.5.2 Orderwire Performance Emulation

The purpose of this section is to illustrate the time required by Network Control Protocol to perform configuration without Virtual Network Configuration using an emulation consisting of Maisie [6] and the actual orderwire code. The emulation of the orderwire systems satisfies several goals. It allows tests of configurations that are beyond the

State	Input Packet	Ouput Packet	Next State	Label
Initialization	None	**MYCALL**	Initialization	I1
	MYCALL	**NEWSWITCH**	Initialization	I2
	NEWSWITCH	**SWITCHPOS**	Active	I3
(Master Only)	timeout (T)	**TOPOLOGY**	Active	I4
	TOPOLOGY	None	Active	I4
Active	**USER_POS**	None	RnUpdate	A1
	TOPOLOGY	None	Active	A2
	MYCALL	None	Initialization	A3
	SWITCHPOS	None	Initialization	A4
RnUpdate	None	None	Active	R1
	None	**HANDOFF**	Active	R2

Table 2.3: Edge Switch Finite State Machine.

State	Input Packet	Ouput Packet	Next State	Label
Initialization	None	None	Active	C1
Active	**timeout**	**USER_POS**	Active	A1
	HANDOFF	None	Initialization	A2

Table 2.4: Remote Node Finite State Machine.

Packet Types	Packet Contents
MYCALL	Callsign, Start-Up-Time
NEWSWITCH	*empty packet*
SWITCHPOS	Global Positioning System Time, Global Positioning System Position
TOPOLOGY	Callsigns and Positions of each node
USER_POS	Callsign, Global Positioning System Time, Global Positioning System Position
HANDOFF	Frequency, Time Slot, Edge Switch Global Positioning System Position

Table 2.5: Network Control Protocol Packets.

scope of the prototype Rapidly Deployable Radio Network hardware. Specifically, it verifies the correct operation of the Rapidly Deployable Radio Network Network Control Protocol in a wide variety of situations. As an additional benefit, much of the actual orderwire code was used with the Maisie emulation allowing further validation of that code. The Edge Switch and Remote Node are modeled as a collection of Maisie entities. This is an emulation rather than a simulation because the Maisie code is linked with the working orderwire code and also with the topology algorithm. There is a Maisie entity for each major component of the Rapidly Deployable Radio Network system including the Global Positioning System receiver, packet radio, inter-Edge Switch links, Remote Node to Edge Switch links and the Master Edge Switch, and Remote Node network configuration processors, as well as other miscellaneous entities. The input parameters to the emulation are shown in Tables 2.6, 2.7, and 2.8.

Parameter	**Definition**
NumRN	Number of Remote Nodes
NumES	Number of Edge Switches
ESDist	Inter Edge Switch spacing (forms rectangular area)
T	ES/ES MYCALL configuration time
maxV	Maximum Remote Node speed for uniform distribution
S	Time to wait between node initial startups
ESspd	Initial Edge Switch speed
ESdir	Initial Edge Switch direction
RNspd	Initial Remote Node speed
RNdir	Initial Remote Node direction

Table 2.6: NCP Emulation Mobility Input Parameters.

Parameter	**Definition**
EndTime	Emulation end time (tenths of seconds)
VCCallTime	Inter High Speed Connection Setup Times
VCCallDuration	High Speed Connection Life Times

Table 2.7: NCP Emulation Time Input Parameters.

Parameter	**Definition**
UseRealTopology	Connect to MatLab and run actual program
Rlink	Maximum beam distance
Fmax	Number of non-interfering frequency pairs
Imult	Interference multiplier
Twidth	Transmitting Beam width
Rwidth	Receiving Beam width

Table 2.8: NCP Emulation Beam Input Parameters.

The time between Virtual Circuit requests from an Remote Node for connections over the inter-Edge Switch antenna beams is assumed to be Poisson. This represents Asynchronous Transfer Mode Virtual Circuit usage over the physical link. The Remote Node will maintain a constant speed and direction until a hand-off occurs, then a new speed and direction are generated from a uniform distribution. This simplifies the analytical computation. Note that Network Control Protocol packet transfer times as measured in Table 2.12 are used here.

The architecture for the Rapidly Deployable Radio Network link management and control is shown in Figure 2.8. The topology module executes only on the Master Edge Switch node. The remaining modules are used on all Edge Switch nodes and Remote Nodes. The beamform module determines an optimal steering angle for the given number of beams which connects all Remote Nodes to be associated with this Edge Switch. It computes an estimated signal to noise interference ratio (SIR) and generates a table of complex weights which, once loaded, will control the beam formation. The connection table is used by the Adaptive High Level Data Link Protocol and Asynchronous Transfer Mode protocol stacks for configuration via the adaptation manager.

The emulation uses as much of the actual network control code as possible. The packet radio driver and Global Positioning System driver interact with Maisie rather than an actual packet radio and Global Positioning System receiver. The remaining

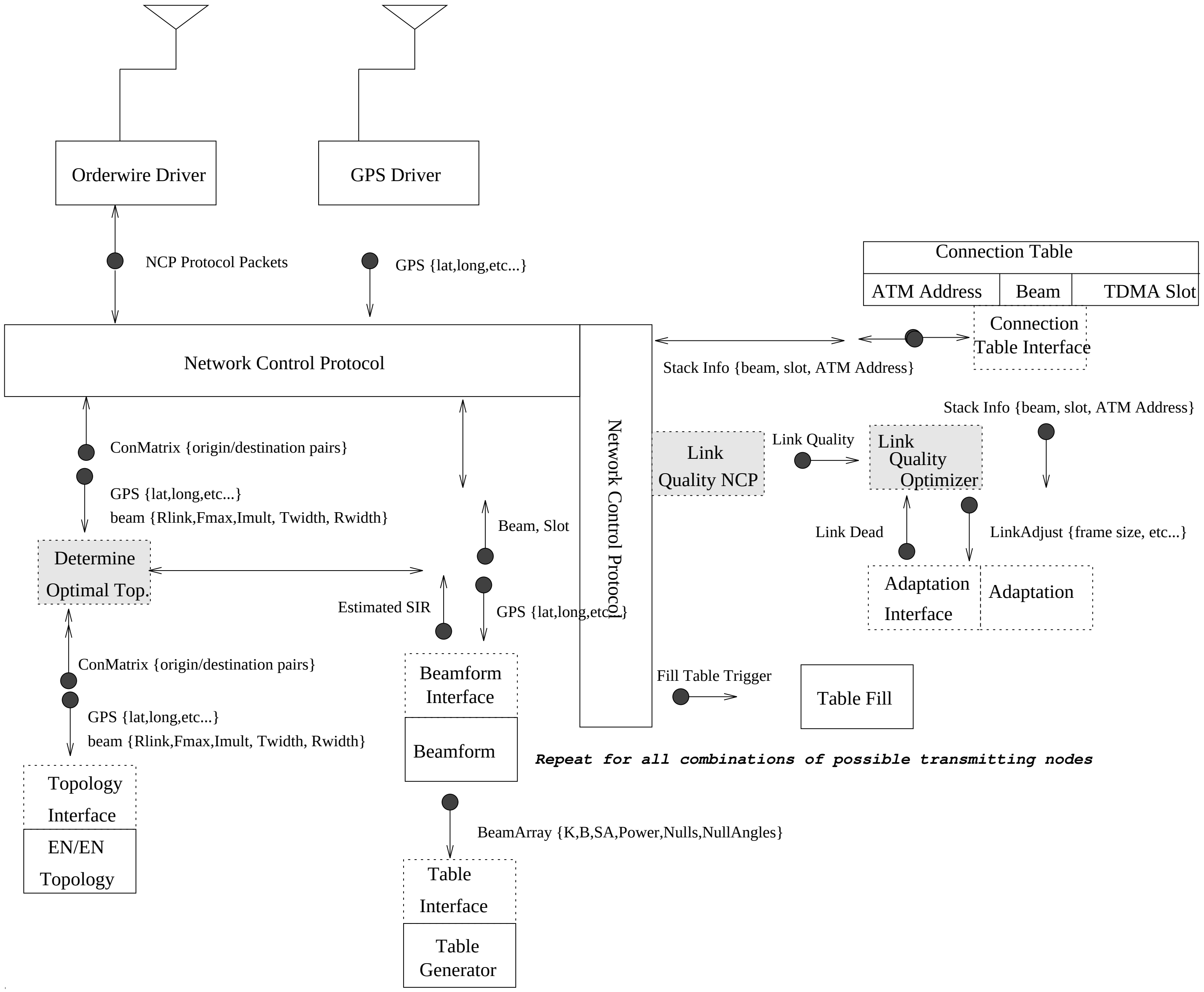


Figure 2.8: Network Control System Architecture.

code is the actual Network Control Protocol code. Figure 2.9 shows the structure of the Maisie entities. The entity names are shown in the boxes and the user defined Maisie message types are shown along the lines. Direct communication between entities is represented as a solid line. The dashed lines indicate from where Maisie entities are spawned. The parent and child entities pass the message types indicated in Figure 2.9.

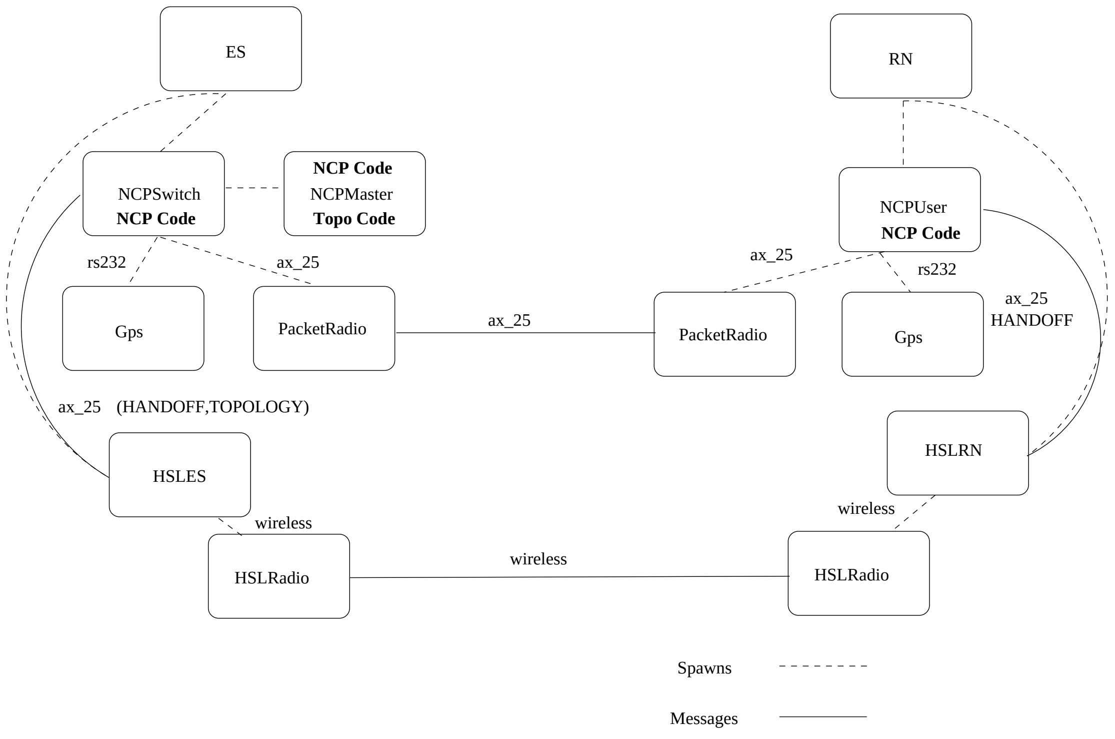


Figure 2.9: Emulation Design.

The Remote Node entity which performs Asynchronous Transfer Mode Virtual Circuit setup generates calls that the Edge Switch node will attempt to accept. The entity labeled HSLRN in Figure 2.9 represents the High Speed Link Remote Node and the entity labeled HSLES in Figure 2.9 represents the High Speed Link Edge Switch entity. These are the radios upon which the Asynchronous Transfer Mode protocol is running. If the Edge Node moves out of range or the Edge Switch has no beam or slot available the setup is aborted. As the Remote Node moves, the Edge Switch will hand off the connection to the proper Edge Switch based on closest distance between Remote Node and Edge Switch.

2.5.3 Emulation Results

This section discusses some of the results from the emulation. Some of these results revealed problems which are not immediately apparent from the Finite State Machine in Tables 2.3 and 2.4. The emulation produces Network Control Protocol Finite State Machine output which shows the transitions based on the Finite State Machine in Tables 2.3 and 2.4. The Finite State Machine output from the simulation provided an easy comparison with diagrams to insure correct operation of the protocol.

Effect of Scale on Network Control Protocol

The emulation was run to determine the effect on the Network Control Protocol as the number of Edge Switch and Remote Node nodes increased. The dominate component of the configuration time is the topology calculation run by the Edge Switch which is designated as the master. Topology calculation involves searching through the problem space of constraints on the directional beams for all feasible topologies and choosing an optimal topology from that set as described in [39]. The units on all values should be consistent with the GPS coordinate units, and all angles are assumed to be degrees. The beam constraint values used in this emulation are: maximum link distance 1000.0 meters, maximum frequencies 3, interference multiplier 1.0, transmit beam width 10.0 degrees, and receive beam width 10.0 degrees. These values were chosen for the emulation before the antenna hardware had been built and the antenna patterns measured.

The topology calculation is performed in MatLab and uses the MatLab provided external C interface. Passing information through this interface is clearly slow, therefore these results do represent the worst case bounds on the execution times of the prototype system.

A speedup will arise through the use of Virtual Network Configuration, which will provide a mechanism for predicting values and also allows processing to be distributed. Another improvement which may be considered is to implement a hierarchical configuration. The network is partitioned into a small number of clusters of nodes in such a way that nodes in each group are as close together as possible. The topology code is run as though these were individual nodes located at the center of each group. This inter-group connection will be added as constraints to the topology computation for the intra-group connections. In this way the topology program only needs to calculate small numbers of nodes which it does relatively quickly.

MYCALL Timer

The MYCALL Timer, set to a value of T, controls how long the system will wait to discover new Edge Switch nodes before completing the configuration. If this value is set too low, new **MYCALL** packets will arrive after the topology calculation has begun, causing the system to needlessly reconfigure. If the MYCALL Timer value is too long, time will be wasted, which will have a large impact on a mobile Edge Switch system. Table 2.9 shows the input parameters and Figure 2.10 shows the time required for all **MYCALL** packets to be received as a function of the number of Edge Switch nodes. These times are the optimal value of the MYCALL Timer as a function of the number of ES nodes because the these times are exactly the amount of time required for all Edge Switch nodes to respond. This time does not include the topology calculation time which is currently significantly longer. Clearly performance would benefit if these time consuming operations could be precomputed. In order to prevent the possibility of an infinite loop of reconfigurations from occurring, an exponential increase in the length of the **MYCALL** Timer value is introduced. As **MYCALL** packets arrive after T has expired, the next configuration occurs with an increased value of T.

Parameter	**Value**
Number of RNs	0
Number of ESs	2 thru 6
Inter-ES Distance	20 m (65.62 ft)
T	20 s
Maximum Velocity	5 m/s (11.16 mi/hr)
Initial Edge Switch Speed	0 m/s
Initial Edge Switch Direction	0^o
Initial Remote Node Speed	N/A
Initial Remote Node Direction	N/A
Call Inter-Arrival Time	1200 s
VC Call Duration	600 s

Table 2.9: MYCALL Timer Simulation Parameters.

Link Usage Probability

Multiple Remote Nodes may share a single beam using Time Division Multiplexing (TDMA) within a beam. The time slices are divided into slots, thus a $(beam, slot)$ tuple defines a physical link. The emulation was run to determine the probability distribution of links used as a function of the number of Remote Nodes. The parameters used in the emulation are shown in Table 2.10 the results of which indicate the number of links and thus the number of distinct $(beam, slot)$ tuples required. Figure 2.11 shows the link usage cumulative distribution function for 4 and 7 Remote Nodes. The number of $(beam, slot)$ combinations in use clearly increases with the number of Remote Nodes as expected.

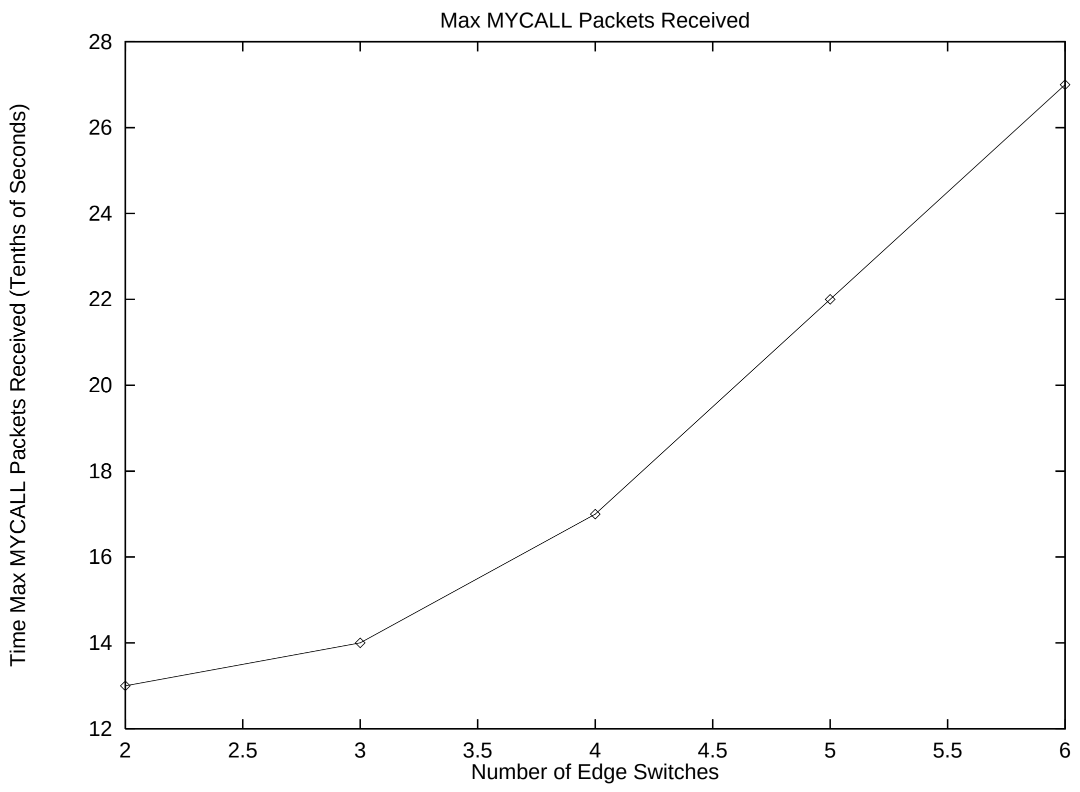


Figure 2.10: MYCALL Packets Received.

Parameter	**Value**
Number of RNs	4 and 7
Number of ESs	2
Inter-ES Distance	20 m (65.62 ft)
T	20 s
Maximum Velocity	5 m/s (11.16 mi/hr)
Initial Edge Switch Speed	0 m/s
Initial Edge Switch Direction	0^o
Initial Remote Node Speed	5 m/s (11.16 mi/hr)
Initial Remote Node Direction	0^o
Call Inter-Arrival Time	1200 s
VC Call Duration	600 s

Table 2.10: Link Usage Simulation Parameters.

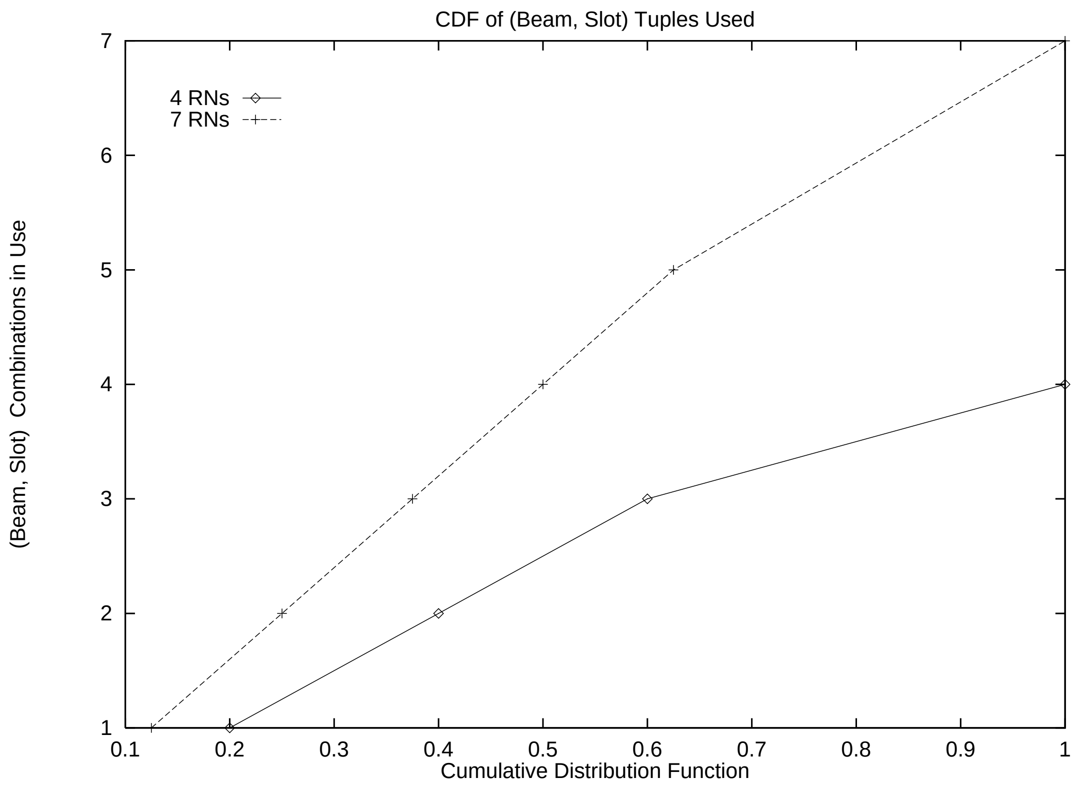


Figure 2.11: $(Beam, Slot)$ Usage.

ES Mobility

ES mobility is a difficult problem because it requires many steps, many of them very time consuming. The parameters used in an emulation with mobile Edge Switch nodes are shown in Table 2.11. As mentioned in the section on the MYCALL Timer, if a **MYCALL** packet arrives after this timer has expired, a reconfiguration occurs. This could happen due to a new Edge Switch powering up or an Edge Switch which has changed position. Figure 2.12 shows the times at which reconfigurations occurred in a situation in which Edge Switch nodes were mobile. Based on the state transitions generated from the emulation it is apparent that the system is in a constant state of reconfiguration; no reconfiguration has time to complete before a new one begins. As Edge Switch nodes move, the Network Control Protocol must notify Remote Nodes associated with an Edge Switch with the new position of the Edge Switch as well as reconfigure the Edge Switch nodes. To solve this problem, a tolerance, which may be associated with the link quality, will be introduced which indicates how far nodes can move within in a beam before the beam angle must be recalculated, which will allow more time between reconfigurations. It is expected that this tolerance in addition to Virtual Network Configuration will provide a solution to this problem.

Effect of Communication Failures

The emulation was run with a given probability of failure on each Network Control Protocol packet type. The following results are based on the output of the finite state machine (FSM) transition output of the emulation and an explanation is given for each case.

A dropped **MYCALL** packet has no effect as long as at least one of the **MYCALL** packets from each Edge Switch is received at the master ES. This is the only use of

Parameter	**Value**
Number of RNs	0
Number of ESs	3
Inter-ES Distance	20 m (65.62 ft)
T	20 s
Maximum Velocity	5 m/s (11.16 mi/hr)
Initial Edge Switch Speed	1 m/s (2.23 mi/hr)
Initial Edge Switch Direction	0^o
Initial Remote Node Speed	N/A
Initial Remote Node Direction	N/A
Call Inter-Arrival Time	1200 s
VC Call Duration	600 s

Table 2.11: Mobile ES Simulation Parameters.

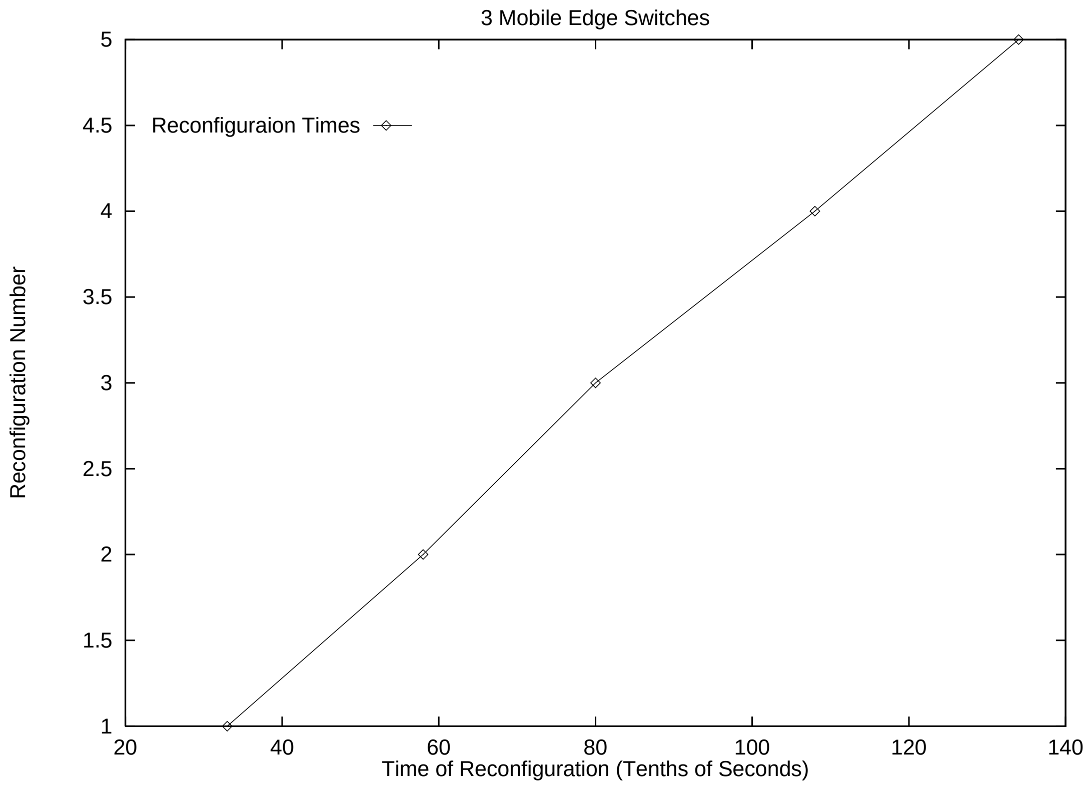


Figure 2.12: Mobile Edge Switch Configuration Time.

the AX.25 broadcast mode in the Edge Switch configuration. The broadcast AX.25 mode is a one time, best effort delivery; therefore, **MYCALL** packets are repeatedly broadcast at the NCP layer.

The Maisie emulation demonstrated that a dropped **NEWSWITCH** packet caused the protocol to fail. This is because the master Edge Switch will wait until it receives all **SWITCHPOS** packets from all Edge Switch nodes for which it had received **MYCALL** packets. The **NEWSWITCH** packet is sent over the AX.25 in connection-oriented mode, e.g. a mode in which corrupted frames are retransmitted. A dropped **SWITCHPOS** packet has the same effect as a dropped **NEWSWITCH** packet. In order to avoid this situation, the Network Control Protocol will re-send the **NEWSWITCH** if no response is received.

Finally, the Maisie emulation showed that a lost **TOPOLOGY** packet results in a partitioned network. The Edge Switch which fails to receive the **TOPOLOGY** packet is not joined with the remaining Edge Switch nodes; however, this Edge Switch node continued to receive and process **USER_POS** packets from all Remote Nodes. It therefore attempts to form an initial connection with all Remote Nodes. The solution for this condition is not to allow Remote Node associations with an Edge Switch node until the **TOPOLOGY** packet is received. Because **MYCALL** packets are transmitted via broadcast AX.25, each Edge Switch node can simply count the number of **MYCALL** packets and estimate the time for the master Edge Switch node to calculate the topology using the number of **MYCALL** packets as an estimate for the size of the network. If no **TOPOLOGY** packet is received within this time period, the Edge Switch node retransmits its **SWITCHPOS** packet to the master Edge Switch node in order to get a **TOPOLOGY** packet as a reply.

This section has provided a background on mobile wireless telecommunications and Rapidly Deployable Radio Network (RDRN). In particular this section has focused on the Network Control Protocol (NCP). Virtual Network Configuration (VNC) will enhance the Network Control Protocol to provide speedup in configuration. The next section introduces the parallel simulation algorithms from which Virtual Network Configuration has been derived.

2.6 Application of Parallel Simulation to Virtual Network Configuration

A general solution to the challenge of mobile wireless Asynchronous Transfer Mode network configuration has been developed in the VNC algorithm. This algorithm is based on concepts for distributed and parallel simulation, which are classified in this section.

Figure 2.13 shows a classification of parallel simulation taken from [67]. In application distribution, the simulation is run independently with different parameter settings. In function level distribution, multiple processors work together on a single simulation execution.

Function level distribution can be further broken down into model function and support function distribution. In support function distribution, support functions such as random number generation are run on separate processors. In model distribution, individual simulation events are distributed on separate processors.

The model distribution class can be further divided into synchronous, asynchronous (space division), and time division approaches. In the synchronous approach, events with the same timestamp are executed in parallel. The asynchronous approach allows greater parallelism to occur. The model is partitioned into logical processes which execute without regard to a common clock. The time division approach partitions the

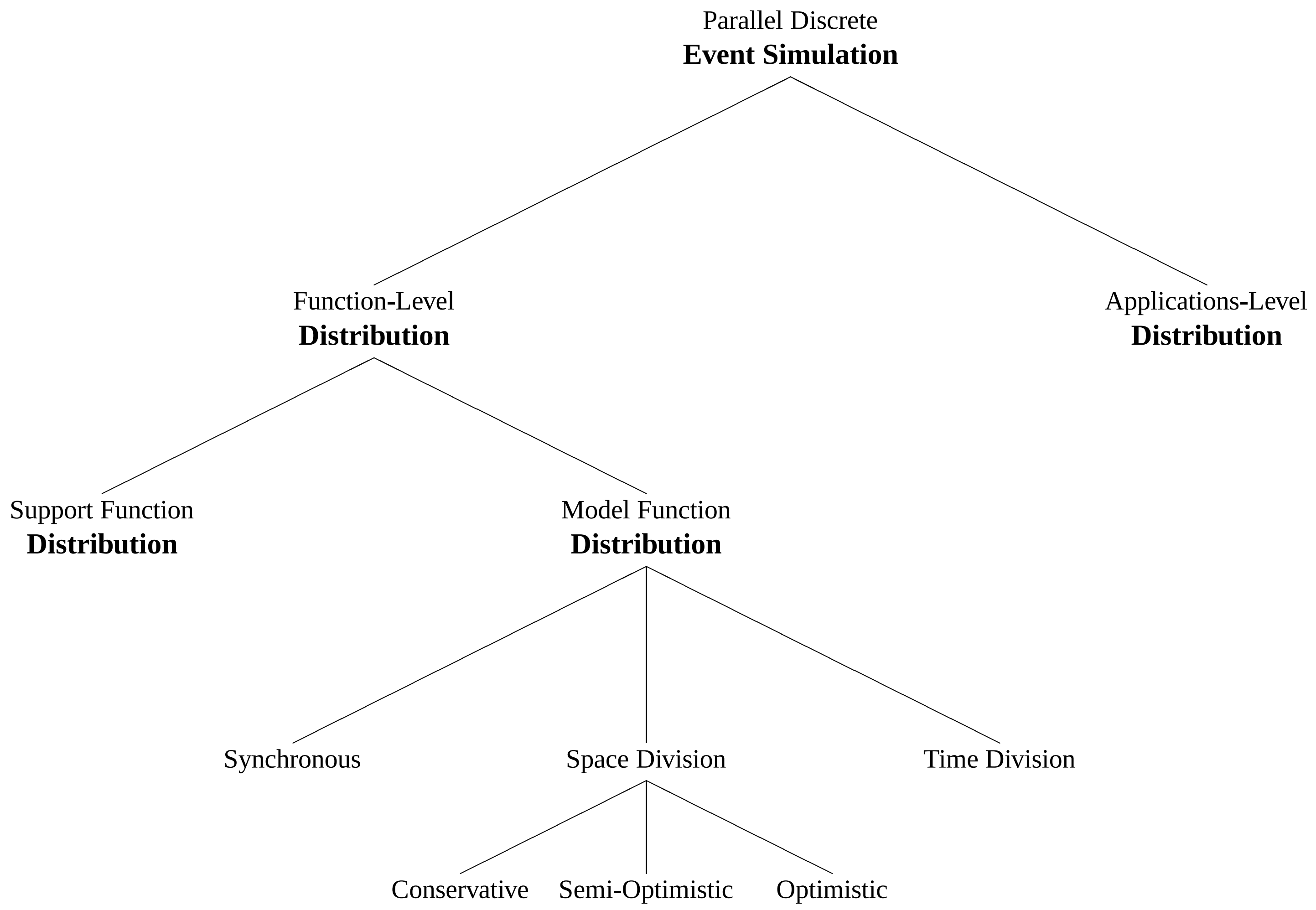


Figure 2.13: Overview of Distributed Simulation Methods.

model in simulation time. Each time period is executed on a separate processor; an estimate is made for the input at times not starting at zero. When the simulation completes, if results of consecutive time partitions do not match, a fix-up phase is required. The asynchronous class is examined in more detail in Section 3.1.

2.7 Mobile Computing and Distributed Simulation

The asynchronous approach is of most interest in this project because it has the capability of taking the most advantage of parallelism. This method can be divided into optimistic, semi-optimistic, and conservative methods. These divisions reflect the amount of deviation allowed from a purely sequential simulation. Optimistic methods allow the most deviation from a sequential simulation and are thus able to take the most advantage of parallelism. The price paid for this is possible out of order message arrival which is corrected in by a rollback mechanism described in detail later. It is this rollback mechanism combined with connection-oriented, predictive mobile network mechanisms which enables the development of Virtual Network Configuration. Changes to a local portion of the Virtual Network Configuration enhanced system are predicted and the results are propagated throughout the system allowing information needed by other elements of the system to be pre-computed and cached.

There are several benefits to be gained from the application of optimistic distributed simulation techniques to network configuration in a mobile wireless environment. Allowing the configuration processes to work ahead, generate results, and cache them for future use provides speedup provided that the predictions are accurate. As an additional benefit, optimistic distributed simulation has a potential to better utilize inherent parallelism thus speeding up operations. The parallelism referred to in this study is among processors currently existing among multiple Rapidly Deployable Radio Network nodes. However, by using Virtual Network Configuration, predictive processes would already exist to take advantage of multiple processors if such processors were added in the future. Optimistic distributed simulation has another characteristic known as super-criticality. The time required for a configuration result to be generated in a system partitioned into Logical Processs is measured; the longest path through which messages propagate is the bottleneck or “critical path”. Super-criticality results from the observation that with the lazy evaluation optimization, it is possible for the final result to be obtained in a time less than the critical path time.

2.8 Virtual Network Configuration Applications

Before going into further detail on Virtual Network Configuration in the next section, consider how a predictive mechanism can be used in a mobile wireless Asynchronous Transfer Mode network. The following sections discuss application of Virtual Network Configuration at various layers of a wireless network. This section clearly demonstrates the potential value for the Virtual Network Configuration algorithm developed in this thesis.

2.8.1 Rapid Physical Layer Setup

A virtual handoff message causes physical layer states to be calculated and cached prior to an actual handoff. Thus the only time required is the time to access the cached information when a handoff occurs. An example of this would be beamsteering calculations in the Rapidly Deployable Radio Network Project [17].

2.8.2 Rapid Media Access Control (Media Access Control) Layer Setup

Adaptive parameters in the Media Access Control Layer will benefit from Virtual Network Configuration. For example the Adaptive HDLC layer [17] in Rapidly Deployable Radio Network will be able to pre-determine parameters such as the optimal frame length, modulation, and coding in advance of a handoff.

2.8.3 Rapid Asynchronous Transfer Mode Topology Calculation

Knowledge of the future location of a mobile host allows the Asynchronous Transfer Mode topology processes to configure prior to handoff taking place. Consider the DEC Autonet [93] [98] which has switch specific features for topology calculation such as Resilient Virtual Circuit calculation and automatic topology calculation. These calculations can be time consuming. Consider the Autonet topology process which takes on the order of $58 + 3.34d + 1.36n + 0.315dn$ milliseconds to determine the network topology, where d is the network diameter and n is the number of nodes. A 100 switch network could take over 0.5 seconds to reconfigure. This means the entire network would be down for half a second each time there was a handoff. In the best case using Virtual Network Configuration, the entire topology calculation is done with virtual messages and the only real-time component would be the time required to update the topology from the Virtual Network Configuration cache. Assuming this takes approximately 5 milliseconds, the speedup for a 100 node network would be 108 times faster with Virtual Network Configuration.

2.8.4 Rapid Asynchronous Transfer ModeARP Server Updates and Configuration

The Asynchronous Transfer Mode Address Resolution Protocol operation is described in [49]. The purpose of Asynchronous Transfer Mode Address Resolution Protocol is to provide the Asynchronous Transfer Mode address for a given network address. The configuration of Asynchronous Transfer Mode hosts and Address Resolution Protocol servers is considered implementation dependent. Virtual Network Configuration will allow the mobile host to be supplied with the location of the Asynchronous Transfer Mode Address Resolution Protocol server within its predicted new Logical IP Subnet, before handoff takes place. This saves the time taken by any implementation dependent protocol from determining the Address Resolution Protocol server during handoff.

2.8.5 Rapid NHRP Server Configuration and Updates

The Next Hop Resolution Protocol (NHRP) [95] allows Asynchronous Transfer Mode Address Resolution Protocol to take place across Logical IP Subnets (LIS). The configuration of NHRP servers may take place via NHRP Registration Packets or manual configuration. NHRP servers can cache the expected changes and have them take effect while the handoff is in progress, giving the appearance of immediate update.

2.8.6 Rapid IP Route Calculation

IP routing updates can take place prior to handoff, with results cached until handoff actually occurs. In [18], handoff timing measurements were taken on a mobile host connected to a 2 Mb/s WaveLAN product by NCR. The stationary hosts connected to a 10 Mb/s Ethernet network. The hosts and base stations were i486 processors with

330 MB hard disks, 16 MB of memory and they ran BSD-Tahoe TCP with the Mobile IP software from Columbia University. Even with instant notification of handoff provided when the mobile host crosses a cell boundary, it took approximately 0.05 seconds for the mobile host routing tables to update, and 0.15 seconds for the mobile support station routing tables to update. Virtual Network Configuration could prepare the tables for update before handoff, and assuming a 5 milliseconds to update the routing table from a local cache, this results in a speedup of 30 times over non-Virtual Network Configuration configuration.

2.8.7 Rapid Mobile-IP Foreign Agent Determination

Mobile-IP configuration results can be cached ahead of time. In the Rapidly Deployable Radio Network Project in particular, the foreign agent information is sent by the Network Configuration Protocol (NCP). This results in a savings of approximately 0.5 seconds as shown in Table 2.12 [15]. These initial measurements were made on Sun Sparcs connected to Kantronics Terminal Node Controller. The experiments involved determining the time required to transmit and process each of the packet types listed in Table 2.5 using the Rapidly Deployable Radio Network packet radios. These times represent the time to packetize, transmit, receive, and depacketize each packet by the Network Control Protocol process. Figure 2.14 illustrates the physical configuration used for the experiments involving the real packet radios. The results are presented in Table 2.12. Most of the overhead occurs during the initial system configuration which occurs only once as long as Edge Switchs remain stationary. With regard to a handoff, the 473 millisecond time to transmit and process the handoff packet is on the same order of time as that required to compute the beam angles and steer the beams. The following sections provide an analysis and discuss the impact of scaling up the system

Event	**Time (ms)**
USER_POS	677
NEWSWITCH	439
HANDOFF	473
MYCALL	492
SWITCHPOS	679
TOPOLOGY	664

Table 2.12: Network Control Protocol Timing Results.

on the configuration time.

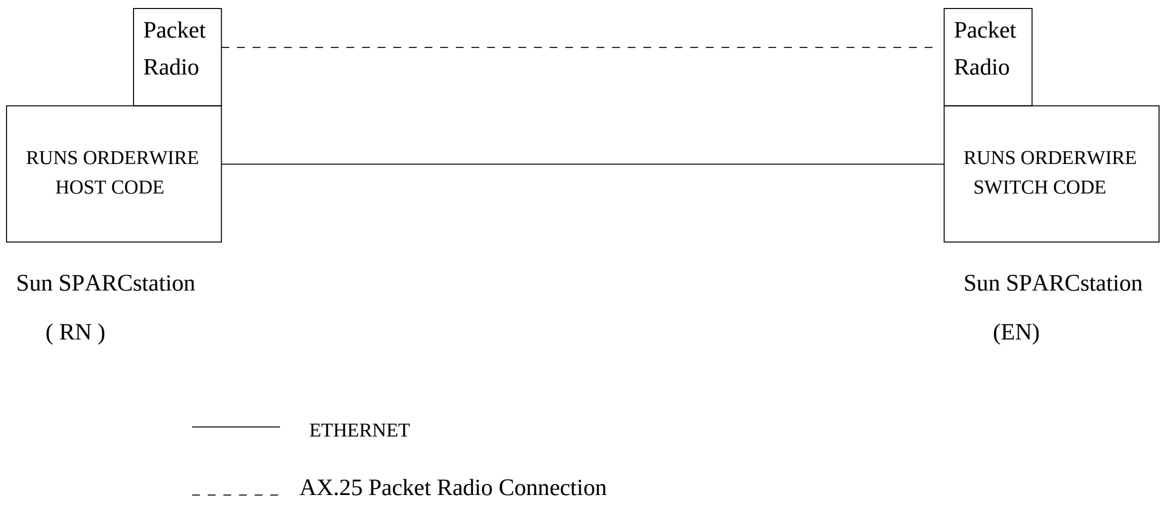


Figure 2.14: Physical Setup for Packet Radio Timing.

2.8.8 Mobile Private Network-Network Interface

Virtual Network Configuration can be used as a basis for a mobile Private Network-Network Interface (Private Network-Network Interface) [22] implementation with minimal changes to the evolving fixed network Private Network-Network Interface standard. Figure 2.15 shows a high level view of the Private Network-Network Interface

Architecture. Terminology used in the Private Network-Network Interface Specification [22] is used here.

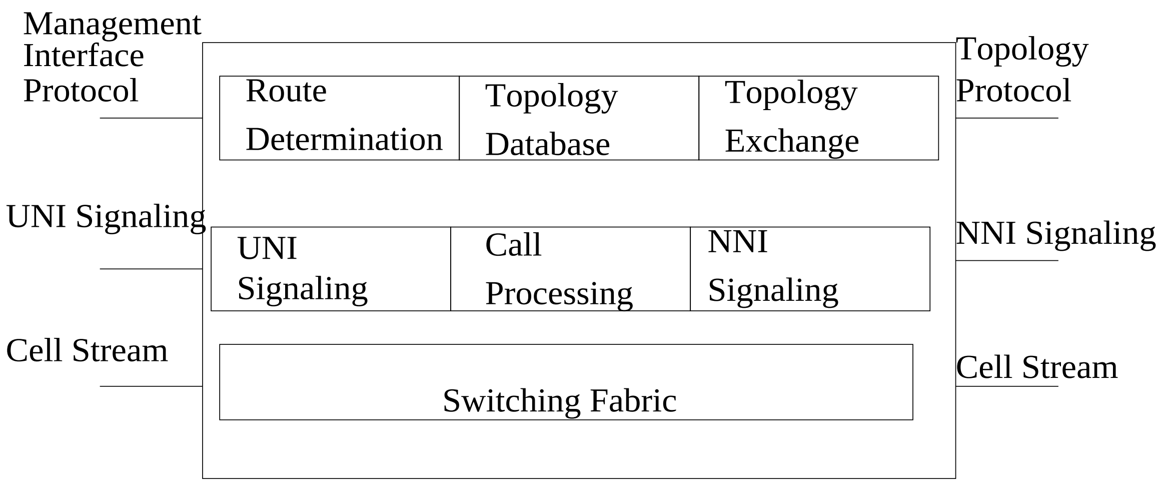


Figure 2.15: Private Network-Network Interface Architecture Reference Model.

In this version of mobile Private Network-Network Interface, the standard Private Network-Network Interface route determination, topology database, and topology exchange would reside within the Network Control Protocol. The NCP stack with Virtual Network Configuration is shown in Figure 2.16.

The enabling mechanism for mobile Private Network-Network Interface is the fact that Virtual Network Configuration will allow the NCP to create a topology which will exist after a hand-off occurs prior to the hand-off taking place. This will cause Private Network-Network Interface to perform its standard action of updating its topology information immediately before the hand-off occurs. Note that this is localized within a single Peer Group. A Peer Group in Private Network-Network Interface is a set of Logical Node grouped together for the purpose of creating a routing hierarchy. Private Network-Network Interface topology messages are flooded among all Logical Nodes in a Peer Group. Note that an Logical Node may be either a lowest level node or a Logical Group Node. A Logical Group Node may consist of many lowest level nodes

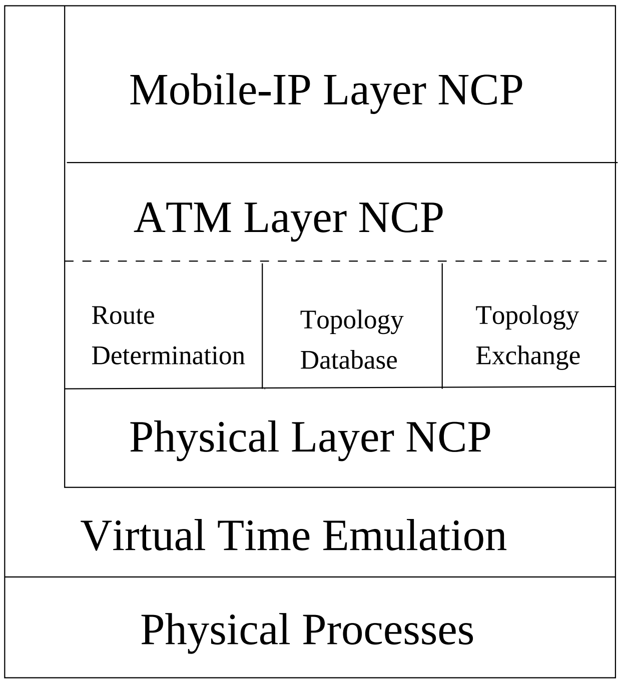


Figure 2.16: Virtual Network Configuration Stack.

treated together as a single node. A Logical Group Node is represented by the Peer Group Leader.

The second enabling mechanism for mobile Private Network-Network Interface is a change to the Private Network-Network Interface signaling protocol. Standard Private Network-Network Interface signaling is allowed to dynamically modify logical links when triggered by a topology change. This will be similar to a **CALL ABORT** Private Network-Network Interface signaling message except that the ensuing Private Network-Network Interface **RELEASE** signaling messages will be contained within the scope of the Peer Group. This will be called a **SCOPED CALL ABORT**. When the topology changes due to an end system hand-off, a check will be made to determine which end system has changed Logical Nodes Logical Node. An attempt is made to establish the same incoming Virtual Circuits at the new Logical Node as were at the original Logical Node and connections are established from the new Logical Node to the original border Logical Nodes of the Peer Group. A border node in Private Network-Network Interface has at least one link which crosses a Peer Group boundary. This allows the end system to continue transmitting with the same Virtual Circuit Identifier as the hand-off occurs. The connections from the original Logical Nodes to the border Logical Nodes are released after the hand-off occurs. If the new Logical Node is already using a Virtual Circuit Identifier that was used at the original Logical Node, the HANDOFF packet will contain the replacement Virtual Circuit Identifiers to be used by the end system. There are now two branches of a logical link tree established with the border Logical Node as the root. After the hand-off takes place the old branch is removed by the **SCOPED CALL ABORT**.

Note that link changes are localized to a single Peer Group. The fact that changes can be localized to a Peer Group greatly reduces the impact on the network and implies that the mobile network should have many levels in its Private Network-Network Interface hierarchy. In order to maintain cell order the new path within the Peer Group is chosen so as to be equal to or longer than the original path based on implementation dependent metrics.

An an example, consider the network shown in Figure 2.17. Peer Groups are enclosed in ellipses and the blackened nodes represent the lowest level Peer Group Leader for each Peer Group. End system A.1.2.X is about to hand-off from A.1.1 to A.2.2. The smallest scope which encompasses the old and new Logical Node is the Logical Node A. A.3.1 is the outgoing border node for Logical Node A. A **CALL SETUP** uses normal Private Network-Network Interface operations to setup a logical link from A.3.1 to A.2.2. After A.1.2.X hands off, a **SCOPED CALL ABORT** releases the logical link from A.3.1 to A.1.1.

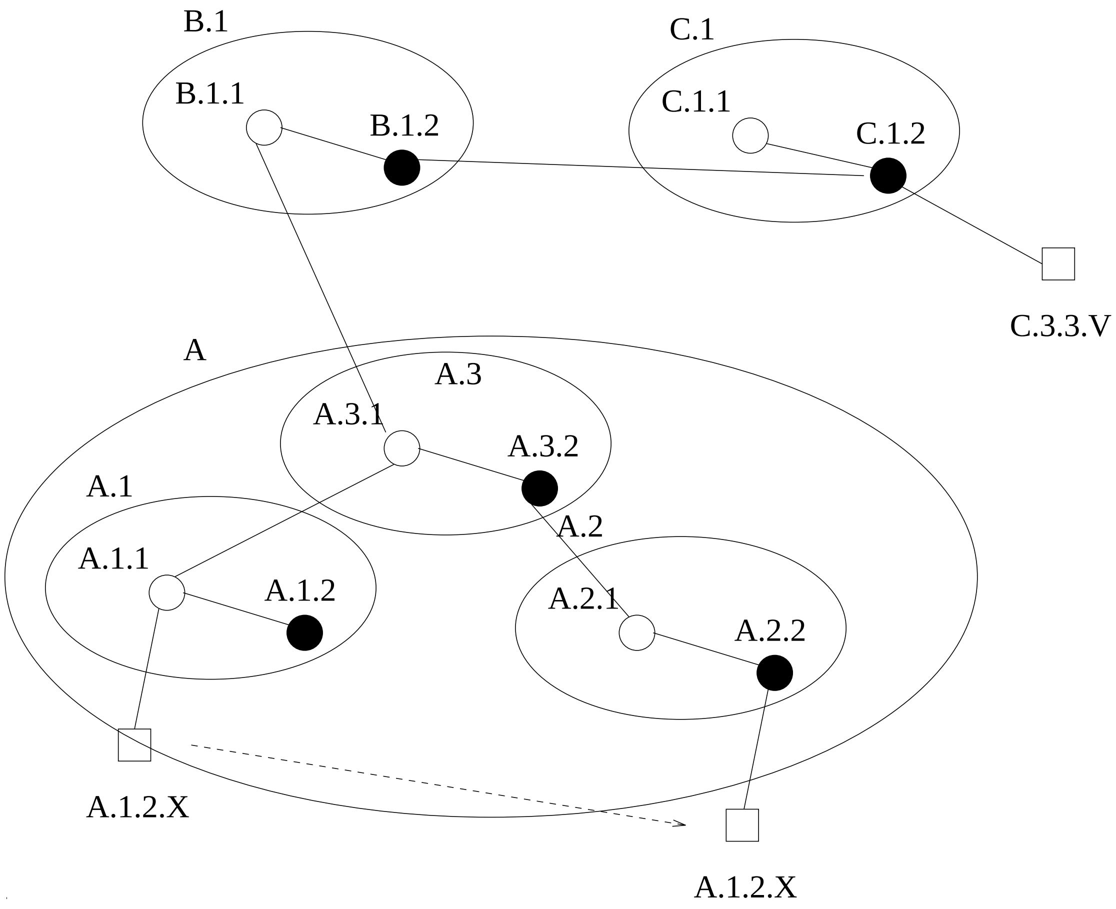


Figure 2.17: Mobile Private Network-Network Interface Example.

2.8.9 Rapid Virtual Circuit Setup with Crankback

An interesting feature of Private Network-Network Interface involves crankback. Crankback is a mechanism for partially releasing a connection setup in progress which has encountered a failure thus allowing the use of alternate routing. This is inherently built-in as a Virtual Network Configuration rollback within the call setup process. The node in the route which detects the link failure will roll back to the time immediately prior to establishing the next hop in the route and as part of the rollback mechanism, causes the predecessor node in the route to rollback and attempt a different route.

2.8.10 Rapid Dynamic Host Configuration

The Dynamic Host Configuration Protocol [27] provides for automatic configuration of a host from data stored on the network. Dynamic Host Configuration Protocol runs over the Asynchronous Transfer Mode link and thus must wait for the link to be established. A Virtual Network Configuration enabled Dynamic Host Configuration Protocol can prepare the contents of the host configuration messages ahead of time.

2.8.11 Eliminate Unnecessary TCP Bandwidth Loss

Knowing that a handoff is about to occur, source traffic can be stopped using the Asynchronous Transfer Mode Available Bit Rate mechanism before the handoff and restarted afterwards. If the state of the TCP timers are preserved, the Van Jacobson TCP recovery mechanism [52], which unnecessarily reduces bandwidth during a handoff, can be avoided.

2.8.12 Rapid Topology Update for Mobile Base Stations

Although base stations which contain Asynchronous Transfer Mode switches generally remain stationary, there is no reason these base stations cannot be mobile. Configuration in this kind of environment is more challenging than a strictly mobile host environment as demonstrated by the emulation results presented in Section 2.5.3 Figure 2.12. Virtual Network Configuration provides a promising solution to this problem by allowing the Edge Switch topology to be precomputed and available for use as soon as the Edge Switch changes position.

2.8.13 Proactive Network Management

The trend in network management is to develop proactive network management systems. These are management systems which ultimately will notify network administration of potential faults before they occur [33]. The interaction between such a predictive network management system and a predictive mobile network is an interesting case to consider.

Imagine a network management station which uses Management Information Base information from either Simple Network Management Protocol or Common Management Information Protocol to construct a model of the network which can use provide alarms for both current and future events. This is an intriguing application of Virtual Network Configuration which is simulated and discussed in Section 5.5. A network management station polls [109] [105] the managed objects for their current status which is a current problem in network management. Polling too frequently wastes bandwidth and degrades the performance of the system being managed. Virtual Network Configuration offers a solution because the predicted events can be used as a guide to control the rate of polling.

Management entities rely upon standard mechanisms to obtain the state of their managed entities in real time. These mechanisms use both solicited and unsolicited methods. The unsolicited method uses messages sent from a managed entity to the manager. These messages are called traps or notifications; the former are not acknowledged while the latter are acknowledged. These messages are very similar to messages used in distributed simulation algorithms; they contain a timestamp and a value, they are sent to a particular destination, i.e. a management entity, and they are the result of an event which has occurred.

Information requested by the management system from a particular managed entity is solicited information. It also corresponds to messages in distributed simulation. It provides a time and a value; however, not all such messages are equivalent to messages in distributed simulation. These messages provide the management station with the current state of the managed entity, even though no event or change of state may have occurred, or more than one event may have occurred. Designing a manager which knows when best to request information is part of our goal.

We will assume for simplicity that each managed entity is represented by a Logical Process (Logical Process). It would greatly facilitate system management if vendors provide not only the standards based Simple Network Management Protocol Management Information Base (Management Information Base), as they do now, but also a standard simulation code which models the entity or application behavior and can be plugged into the management system just as in the case with a Management Information Base.

There are several parameters in this application which must be determined. The first is how often this application should check the Logical Process to verify that past results match reality. This phase is called **Verification Query** (Υ). The optimum choice of verification query interval (T_{query}) is important because querying of entities should be minimized while still guaranteeing accuracy is maintained within some predefined tolerance, Θ. This tolerance could be set for each state variable or message value sent from an Logical Process.

The amount of time into the future which the emulation will attempt to venture is another parameter which must be determined. This lookahead sliding window of width $(t..\Lambda)$ where t is the current time, should be preconfigured based on the accuracy required; the farther ahead the application attempts to go past real time, the more risk that is assumed.

Optimum Choice of Verification Query Times

One method of choosing the optimum verification query time would be to query the entity based on the frequency of the data monitored. Assuming the simulated data is correct, query or sample in such a way as to perfectly reconstruct the data, e.g. based on the maximum frequency component. A possible drawback is that the actual data may be changing at a multiple of the predicted value. The samples may appear to to be accurate when they are invalid. Some interesting points about this are that the error throughout the simulated system may be randomized in such a way that errors among Logical Processs cancel. However, if the simulation is composed of many of the same class of Logical Process, the errors may compound rather than cancel each other. See Section 5.5 on page 177 for network management simulation results.

Predictive System Interaction

There is an interesting interaction between the predictive management system and the predictive mobile network. A predictive mobile network such as the Virtual Network Configuration proposal for Rapidly Deployable Radio Network [15] will have results cached in advance of use for many configuration parameters. These results should be part of the Management Information Base for the mobile network and should include the predicted time of the event which requires the result, the value of the result, and the probability that the result will be correct at that time. Thus there will be a triple associated with each predicted event: *(time, value, probability)*. Network management protocols, e.g. Simple Network Management Protocol [94] and Common Management Information Protocol [50], include the time as part of the Protocol Data Unit, however this time indicates only the real time the poll occurred.

A predictive management system could simply use Logical Processs to represent the predictive mobile processes as previously described, however, this is redundant since the mobile network itself has predicted events in advance as part of its own management and control system. Therefore, managing a predictive mobile network with a predictive network management system provides an interesting problem in trying to get the maximum benefit from both of these predictive systems.

Combining the two predictive systems in a low level manner, e.g. allowing the Logical Processs to exchange messages with each other, raises questions about synchronization between the mobile network and the management station. However, the predicted mobile network results can be used as additional information to refine the management system results. The management system will have computed *(time, value, probability)* triples for each predicted result as well. The final result by the management system would then be an average of both triples weighted by the probability. An additional weight may be added given the quality of either system. For example the network management system might be weighted higher because it has more knowledge about the entire network. Alternatively, the mobile network system may be weighted higher because the mobile system may have better predictive capability for the detailed events concerning handoff. Thus the two systems do not directly interact with each other, but the final result is a combination of the results from both predictive systems.

A more complex method of combining results from these two systems would involve a causal network such as the one described in [65]. Network management data freshness and other issues critical in a mobile environment are examined in [120] and [119].

Chapter 3

VNC Algorithm Description

3.1 The Origin of Virtual Network Configuration

Recently proposed mobile networking architectures and protocols involve predictive mobility management schemes. For example, an optimization to a Mobile IP-like protocol using IP-Multicast is described in [99]. Hand-offs are anticipated and data is multicast to nodes within the neighborhood of the predicted handoff. These nodes intelligently buffer the data so that no matter where the mobile host (MH) re-associates after a handoff, no data will be lost. Another example [70] [71] proposes deploying mobile floating agents which decouple services and resources from the underlying network. These agents would be pre-assigned and pre-connected to predicted user locations.

One of the major contributions of this research is to recognize and define an entirely new branch of the Time Warp Family Tree of algorithms. Virtual Network Configuration integrates real and virtual time at a fundamental level allowing processes to execute ahead in time. The Virtual Network Configuration algorithm must run in real-time, that is, with hard real-time constraints. Consider the work leading towards the predictive Virtual Network Configuration algorithm starting from a classic paper on synchronizing clocks in a distributed environment [62]. A theorem from this paper limits the amount of parallelism in any distributed simulation algorithm:

Rule 1 *If two events are scheduled for the same process, then the event with the smaller timestamp must be executed before the one with the larger timestamp.*

Rule 2 *If an event executed at a process results in the scheduling of another event at a different process, then the former must be executed before the latter.*

A parallel simulation method, known as CMB (Chandy-Misra-Bryant), which predates Time Warp [54] is described in [21]. CMB is a conservative algorithm which uses Null Messages to preserve message order and avoid deadlock. Another method developed by the same author does not require Null Message overhead, but includes a central controller to maintain consistency and detect and break deadlock. There has been much research towards finding a faster algorithm and many algorithms claiming to be faster have compared themselves against the CMB method.

The basic Time Warp Algorithm [54] was a major advance in distributed simulation. Time Warp is an algorithm used to speedup Parallel Discrete Event Simulation by taking advantage of parallelism among multiple processors. It is an optimistic method because all messages are assumed to arrive in order and are processed as soon as possible. If a message arrives out-of-order at an Logical Process, the Logical Process rollbacks back to a state which was saved prior to the arrival of the out-of-order message. Rollback occurs by sending copies of all previously generated messages as anti-messages. Anti-messages are exact copies of the original message, except and anti-bit is set within the field of the message. When the anti-message and real message meet, both messages are removed. Thus, the rollback cancels the effects of out-of-order messages. The rollback mechanism is a key part of Virtual Network Configuration, and algorithms which improve Time Warp and rollback will also improve Virtual Network Configuration. There continues to be an explosion of new ideas and protocols for improving Time Warp. An advantage to using a Time Warp based algorithm is the ability to leverage future optimizations. There have been many variations and improvements to this basic algorithm for parallel simulation. A collection of optimizations to Time Warp is provided in [34]. The technical report describing Time Warp [54] does not solve the problem of determining Global Virtual Time, however an efficient algorithm for the determination of Global Virtual Time is presented in [63]. This algorithm does not require message acknowledgments thus increasing the performance, yet the algorithm works with unreliable communication links.

An analytical comparison of CMB and Time Warp is the focus of [68]. In this paper the comparison is done for the simplified case of feed-forward and feedback networks. Conditions are developed for Time Warp to be conservative optimal. Conservative optimal means that the time to complete a simulation is less than or equal to the critical path [10] through the event-precedence graph of a simulation.

A search for the upper bound of the performance of Time Warp versus synchronous distributed processing methods is presented in [32]. Both methods are analyzed in a feed-forward network with exponential processing times for each task. The analysis in [32] assumes that no Time Warp optimizations are used. The result is that Time Warp has an expected potential speedup of no more than the natural logarithm of P over the synchronous method where P is the number of processors.

A Markov Chain analysis model of Time Warp is given in [38]. This analysis uses standard exponential simplifying assumptions to obtain closed form results for performance measures such as fraction of processed events that commit, speedup, rollback recovery, expected length of rollback, probability mass function for the number of uncommitted processed events, probability distribution function of the local virtual time of a process, and the fraction of time the processors remain idle. Although the analysis appears to be the most comprehensive analysis to date, it has many simplifying assumptions such as no communications delay, unbounded buffers, constant message population, message destinations are uniformly distributed, and rollback takes no time. Thus, the analysis in [38] is not directly applicable to the time sensitive nature of Virtual Network Configuration.

Further proof that Time Warp out-performs CMB is provided in [69]. This is done by showing that there exists a simulation model which out-performs CMB by exactly the number of processors used, but that no such model in which CMB out-performs Time Warp by a factor of the number of processors used exists.

A detailed comparison of the CMB and Time Warp methods are presented in [67]. It is shown that Time Warp out-performs conservative methods under most conditions. Improvements to Time Warp are suggested by reducing the overhead of state saving information and the introduction of a global virtual time calculation. Simulation study results of Time Warp are presented in [114]. Various parameters such as communication delay, process delay, and process topology are varied and conditions under which Time Warp and CMB perform best are determined.

The major contribution of this section is to recognize and define an entirely new branch of the Time Warp Family Tree of algorithms, shown in Figure 3.1, which integrates real and virtual time at a fundamental level. The Virtual Network Configuration algorithm must run in real-time, that is, with hard real-time constraints. Real-time constraints for a time warp simulation system are discussed in [35]. The focus in [35] is the R-Schedulability of events in Time Warp. Each event is assigned a real-time deadline ($d_{E_{i,T}}$) for its execution in the simulation. R-Schedulability means that there exists a finite value (R) such that if each event's execution time is increased by R, the event can still be completed before its deadline. The first theorem from [35] is that if there is no constraint on the number of such false events that may be created between any two successive true events on an Logical Process, Time Warp cannot guarantee that a set of R-schedulable events can be processed without violating deadlines for any finite R. There has been a rapidly expanding family of Time Warp algorithms focused on constraining the number of false events which are discussed next.

Another contribution of this chapter is to classify these algorithms as shown in Figures 3.1, 3.2, 3.3 and Table 3.1. Each new modification to the Time Warp mechanism attempts to improve performance by reducing the expected number of rollbacks. Partitioning methods attempt to divide tasks into logical processes such that the inter-Logical Process communication is minimized. Also included under partitioning are methods which dynamically move Logical Processes from one processor to another in order to minimize load and/or inter-Logical Process traffic. Delay methods attempt to introduce a minimal amount of wait into Logical Processes such that the increased synchronization and reduced number of rollbacks more than compensates for the added delay. Many of the delay algorithms use some type of windowing method to bound the difference between the fastest and slowest processes. The bounded sphere class of delay mechanisms attempt to calculate the maximum number of nodes which may need to be rolled back because they have processed messages out of order. For example, $S \downarrow (i, B)$ in [73] is the set of nodes effected by incoming messages from node i in time B while $S \uparrow (i, B)$ is the set of nodes effected by outgoing messages from node i in time B. The downward pointing arrow in $S \downarrow (i, B)$ indicates incoming messages while the upward pointing arrow in $S \uparrow (i, B)$ indicates outgoing messages. Another approach to reducing rollback is to use all available semantic information within messages. For example, commutative sets of messages are messages which may be processed out-of-order yet they produce the same result. Finally, probabilistic methods attempt to predict certain characteristics of the optimistic simulation, usually based on its immediate past history, and take action to reduce rollback based on the predicted characteristic. It is insightful to review a few of these algorithms because they not only trace the development of Time Warp based algorithms but because they illustrate the "state of the art" in preventing rollback, attempts at improving performance by constraining lookahead, partitioning of Logical Processes into sequential and parallel environments, and the use of semantic information. All of these techniques and more may be applied in the Virtual Network Configuration algorithm.

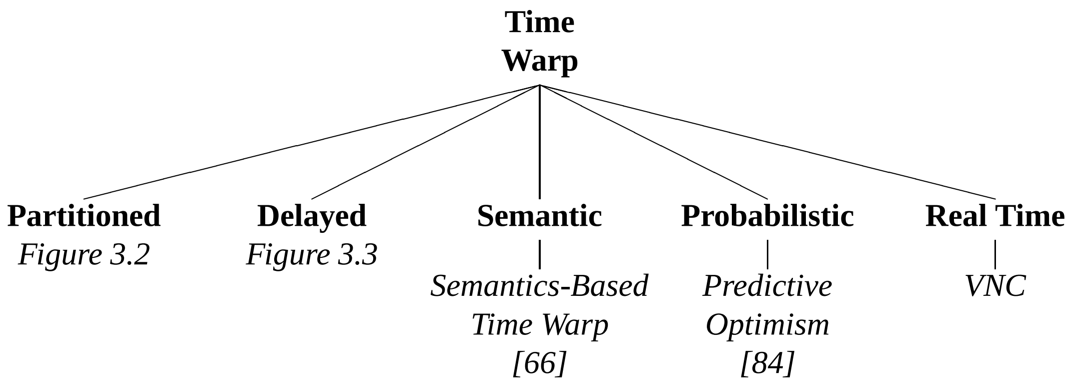


Figure 3.1: Time Warp Family of Algorithms.

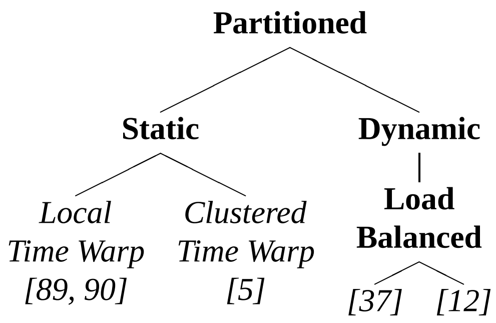


Figure 3.2: Partitioned Algorithms.

The Bounded Lag algorithm [74] for constraining rollback explicitly calculates, for each Logical Process, the earliest time that an event from another Logical Process may

Class	Sub Class	Sub Class	Description	Example
Probabilistic			Predict next msg. arrival time	Predictive Optimism [66]
Semantic			Utilize msg contents to reduce rollback	Semantics Based Time Warp [66]
Partitioned			Inter-LP communication minimized	
	Dynamic		LPs change mode while executing	
		Load Balanced	LPs can migrate across hosts	[37], [12]
	Static		LPs cannot change mode while executing	Clustered Time Warp [5] Local Time Warp [89, 90]
Delayed			Delays added to reduce rollback	
	Windowed		Window based mechanisms reduce rollback	
		Adaptive Window	Window adapts to reduce rollback	Breathing Time Warp [107]
		Fixed Window	Window size does not adapt	Breathing Time Buckets [107] Moving Time Windows [77]
	Bounded Sphere		Based on earliest time inter-LP effects occur	Bounded Lag [74] WOLF [77, 102]
	Non-Windowed		A mechanism other than window to reduce rollback.	Adaptive Time Warp [9] Near Perfect State Information [104]

Table 3.1: Time Warp Family of Algorithms.

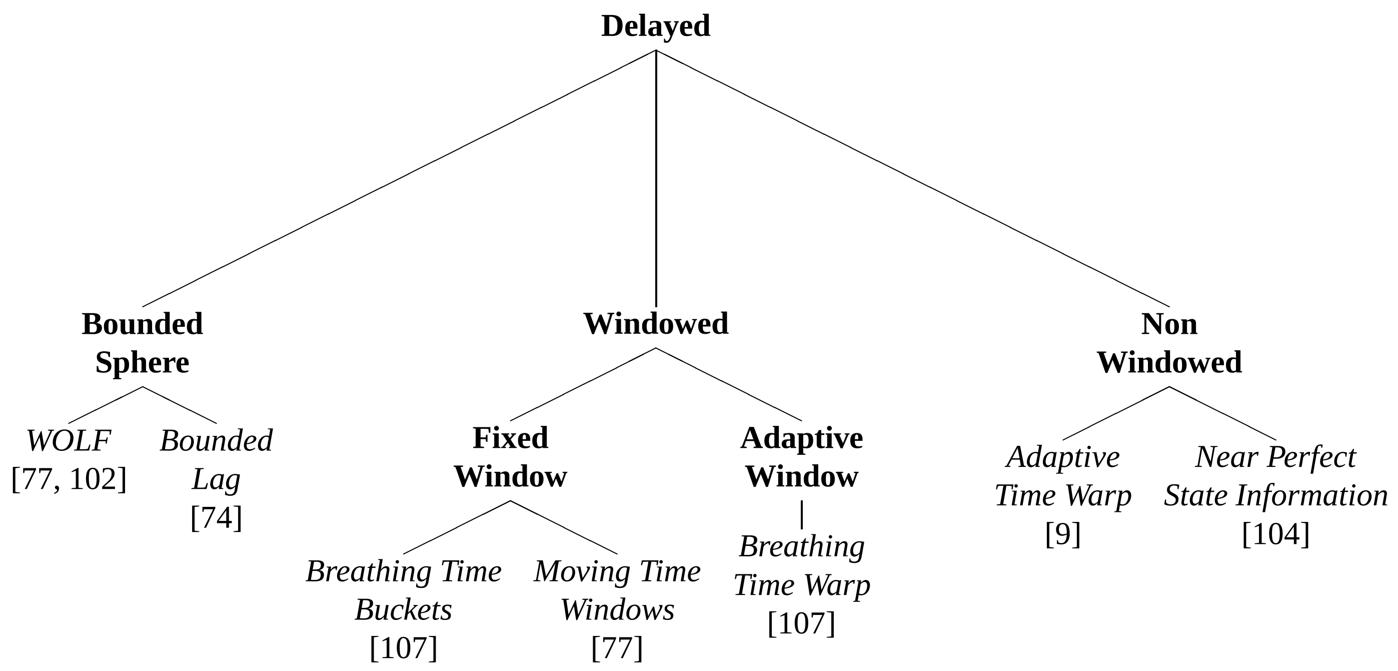


Figure 3.3: Delaying Algorithms.

$$\alpha(i) = \min_{j \in S\downarrow(i,B) \wedge j \neq i} \{d(j,i) + \min\{T(j), d(i,j) + T(i)\}\} \tag{3.1}$$

affect the current Logical Process's future. This calculation is done by first determining the reachability sphere ($S \downarrow (i, B)$) which is the set of nodes which a message may reach in time B. This depends on the minimum propagation delay of a message in simulation time from node i to node j which is $d(i, j)$. Once $S \downarrow (i, B)$ is known, the earliest time that node i can be affected, $\alpha(i)$, is shown in Equation 3.1 where $T(i)$ is the minimum message receive time in node i's message receive queue. After processing all messages up to time $\alpha(i)$, all Logical Processes must synchronize.

The Bounded Lag algorithm is conservative because it synchronizes Logical Processes so that no message arrives out of order. The problem is that a minimum $d(i, j)$ must be known and specified before the simulation begins. A large $d(i, j)$ can negate any potential parallelism, because a large $d(i, j)$ implies a large $\alpha(i)$ which implies a longer time period between synchronizations. A filtered rollback extension to Bounded Lag is described in [73]. Filtered Rollback allows $d(i, j)$ to be made arbitrarily small which may possibly generate out of order messages. Thus the basic rollback mechanism described in [54] is required.

A thorough understanding of rollbacks and their containment are essential for Virtual Network Configuration. In [73], rollback cascades are analyzed under the assumption that the Filtered Rollback mechanism is used. Rollback activity is viewed as a tree; a single rollback may cause one or more rollbacks which branch-out indefinitely. The analysis is based on a "survival number" of rollback tree branches. The survival number is the difference between the minimum propagation delay $d(j, i)$ and the delay in simulated time for an event at node i to effect the history at node j, $t(i, j)$. Each generation of a rollback caused by an immediately preceding node's rollback adds a positive or negative survival number. These rollbacks can be thought of as a tree whose leaves are rollbacks which have "died out". It is shown that it is possible to calculate upper bounds, namely, infinite or finite number of nodes in the rollback tree.

A probabilistic method is described in [84]. The concept in [84] is that optimistic simulation mechanisms are making implicit predictions as to when the next message will arrive. A purely optimistic system assumes that if no message has arrived, then no message **will** arrive and computation continues. However, the immediate history of the simulation can be used to attempt to predict when the next message will arrive. This information can be used either for partitioning the location of the Logical Processes on processors or for delaying computation when a message is expected to arrive.

In [78], a foundation is laid for unifying conservative and optimistic distributed simulation. Risk and aggressiveness are parameters which are explicitly set by the simulation user. Aggressiveness is the parameter controlling the amount of non-causality allowed in order to gain parallelism and risk is the passing of such results through the simulation system. Both aggressiveness and risk are controlled via a windowing mechanism similar to the sliding lookahead window of the Virtual Network Configuration algorithm.

A unified framework for conservative and optimistic simulation called ADAPT is described in [55]. ADAPT allows the execution of a "sub-model" to dynamically change from a conservative to an optimistic simulation approach. This is accomplished by uniting conservative and optimistic methods with the same Global Control Mechanism. The mechanism in [55] has introduced a useful degree of flexibility and described the mechanics for dynamically changing simulation approaches, [55] does not quantify or discuss the optimal parameter settings for each approach.

A hierarchical method of partitioning Logical Processes is described in [89, 90]. The salient feature of this algorithm is to partition Logical Processes into clusters. The

Logical Processes operate as in Time Warp. The individual clusters interact with each other in a manner similar to Logical Processes.

The Clustered Time Warp is described in [5]. The Clustered Time Warp mechanism was developed concurrently but independently of Virtual Network Configuration. This approach uses Time Warp between clusters of Logical Processes residing on different processors and a sequential algorithm within clusters. This is in some ways similar to the Single Processor Logical Process described later in Virtual Network Configuration. Since the partitioning of the simulation system into clusters is a salient feature of this algorithm, Clustered Time Warp has been categorized as a partitioned algorithm in Figure 3.2. One of the contributions of [5] in Clustered Time Warp is an attempt to efficiently control a cluster of Logical Processes on a processor by means of the Clustered Environment. The Clustered Environment allows the Logical Processes to behave as individual Logical Processes as in the basic time warp algorithm or as a single collective Logical Process. The Clustered Time Warp algorithm is an optimization method for the Virtual Network Configuration Single Processor Logical Processes.

Semantics Based Time Warp is described in [66]. In this algorithm, the Logical Processes are viewed as abstract data type specifications. Messages sent to an Logical Process are viewed as function call arguments and messages received from Logical Processes are viewed as function return values. This allows data type properties such as commutativity to be used to reduce rollback. For example, if commutative messages arrive out-of-order, there is no need for a rollback since the results will be the same. This is used in the **MYCALL** Timeout phase of the Network Control Protocol which is transition labeled I4 in Table 2.3 on page 37. The order in which the initial Edge Switch **MYCALL** messages are collected does not impact the result of the configuration.

Another means of reducing rollback, in this case by decreasing the aggressiveness of Time Warp, is given in [9]. This scheme involves voluntarily suspending a processor whose rollback rate is too frequent because it is out-pacing its neighbors. Virtual Network Configuration uses a fixed sliding window to control the rate of forward emulation progress, however, a mechanism based on those just mentioned could be investigated.

The Near Perfect State Information Adaptive Synchronization Algorithms for Parallel Discrete Event Simulation are discussed in [104] and [103]. The adaptive algorithms use feedback from the simulation itself in order to adapt. Some of the deeper implications of these types of systems are discussed in Appendix E. The NPSI system requires an overlay system to return feedback information to the Logical Processes. The Near Perfect State Information Adaptive Synchronization Algorithm examines the system state (or an approximation of the state) calculates an error potential for future error, then translates the error potential into a value which controls the amount of optimism.

Breathing Time Buckets described in [106] is one of the simplest fixed window techniques. If there exists a minimum time interval between an each event and the earliest event generated by that event (T), then the system runs in time cycles of duration T. All Logical Processes synchronize after each cycle. The problem with this approach is that T must exist and be known ahead of time. Also, T should be large enough to allow a reasonable amount of parallelism, but not so large as to loose fidelity of the system results.

Breathing Time Warp ([107]) attempts to overcome the problems with Breathing Time Buckets and Time Warp by combining the two mechanisms. The simulation mechanism operates in cycles which alternate between a Time Warp phase and a Breathing Time Buckets phase. The reasoning for this mechanism is that messages close to Global Virtual Time are less likely to cause a rollback while messages with time-stamps far from Global Virtual Time are more likely to cause rollback. Breathing Time Warp also introduces the *event horizon* which is the earliest time of the next new event generated in the current cycle. A user controlled parameter controls the number of messages which are allowed to be processed beyond Global Virtual Time. Once this number of messages is generated in the Time Warp phase, the system switches to the Breathing Time Buckets phase. This phase continues to process messages, but does not send any new messages. Once the event horizon is crossed, processing switches back to the Time Warp phase. One can picture the system taking in a breath during the Time Warp phase and exhaling during the Breathing Time Buckets phase.

An attempt to reduce roll-backs is presented in an algorithm called WOLF [77, 102]. This method attempts to maintain a sphere of influence around each rollback in order to limit its effects.

The Moving Time Window [101, 102] simulation algorithm is an interesting alternative to Time Warp. It controls the amount of aggressiveness in the system by means of a moving time window Moving Time Windows. The trade-off in having no roll-backs in this algorithm is loss of fidelity in the simulation results. This could be considered as another method for implementing the Virtual Network Configuration algorithm.

An adaptive simulation application of Time Warp is presented in [113]. The idea presented in this paper is to use Time Warp to change the input parameters of a running simulation without having to restart the entire simulation. Also, it is suggested that events external to the simulation can be injected even after that event has been simulated.

Hybrid simulation and real system component models are discussed in [7]. The focus in [7] is on Partially Implemented Performance Specification Components of a performance specification for a distributed system are implemented while the remainder of the system is simulated. More components are implemented and tested with the simulated system in an iterative manner until the entire distributed system is implemented. The Partially Implemented Performance Specification system described in [7] discusses using MAY or Maisie as a tool to accomplish the task, but does not explicitly discuss Time Warp.

The work in [35] provides some results relevant to Virtual Network Configuration. It is theorized that if a set of events is R-schedulable in a conservative simulation, where $R \geqslant \rho + ct + \sigma$ where ρ is the time to restore an Logical Process state, c is the number of Logical Processes, t is the time the simulation has been running, and σ is the real time required to save an Logical Process state, then it can run to completion without missing any deadline by an No False Time-stamps Time Warp strategy with lazy cancellation. No False Time-stamps Time Warp assumes that if an incorrect computation produces and incorrect event ($E_{i,T}$), then it must be the case that the correct computation also produces an event ($E_{i,T}$) with the same timestamp [1]. This result shows that conditions exist in a Time Warp algorithm which guarantee that events are able meet a given deadline. This is encouraging for the Virtual Network Configuration algorithm since clearly events must be completed before real-time reaches the predicted time of the event for the cached results to be useful in Virtual Network Configuration. Finally, this author has not been the only one to consider the use of Time Warp to speedup a real-time process. In [110], the idea of temporal decoupling is applied to a signal processing environment. Differences in granularity of the rate of execution are utilized to cache results before needed and to allocate resources more effectively.

This section has shown the results of research into improving Time Warp, especially in reducing rollback, as well as the limited results in applying Time Warp to real time systems. Improvements to Time Warp and the application to real time systems are both directly applicable to Virtual Network Configuration. Now consider the Virtual Network Algorithm in more detail.

[1]This simplification makes the analysis in [35] tractable. This assumption also greatly simplifies the analysis of Virtual Network Configuration. The Virtual Network Configuration algorithm is simplified because the state verification component of Virtual Network Configuration requires that saved states be compared with the real-time state of the process. This is done easily under the assumption that the T (timestamp) values of the two events E_{i,T_v} and E_{i,T_r} are the same.

3.2 Virtual Network Configuration Description

New terminology is introduced in the description of the Virtual Network Configuration algorithm which follows and terminology borrowed from previous distributed simulation algorithm descriptions has a slightly different meaning in Virtual Network Configuration, thus it is important that terms be precisely defined.

The Virtual Network Configuration algorithm encapsulates each Physical Process within an Logical Process. A Physical Process is nothing more than an executing task defined by program code. An example of a Physical Process is the Rapidly Deployable Radio Network beam table computation task. The beam table computation task generates a table of complex weights which controls the angle of the radio beams based on position input. An Logical Process consists of the Physical Process and additional data structures and instructions which maintain message order and correct operation as the system executes ahead of real time. As an example, the beam table computation Physical Process is encapsulated in an Logical Process which maintains generated beam tables in its State Queue and handles rollback due to out-of-order input messages or out-of-tolerance real messages as explained later. An Logical Process contains a Receive Queue, Send Queue (QS), and State Queue (SQ). The Receive Queue maintains newly arriving messages in order by their Receive Time (TR). The QS maintains copies of previously sent messages in order of their send times. The state of an Logical Process is periodically saved in the SQ. The Logical Process also contains its notion of time known as Local Virtual Time (LVT) and a Tolerance (Θ). Tolerance is the allowable deviation between actual and predicted values of incoming messages. For example, when a real message enters the beam table computation Logical Process the position in the message value is compared with the position which had been cached in the State Queue of the Logical Process. If these values are out of tolerance, then corrective action is taken in the form of a rollback as explained later. Also, the Current State (CS) of an Logical Process is the current state of the structures and Physical Process encapsulated within an Logical Process.

The Virtual Network Configuration system contains a notion of the complete system time known as Global Virtual Time (GVT) and a sliding window of length Look Ahead time (Λ). There have been several proposals for efficient determination of Global Virtual Time, for example [63]. The Global Virtual Time algorithm in [63] allows Global Virtual Time to be determined in a message-passing environment as opposed to the easier case of a shared memory environment. Virtual Network Configuration allows only message passing communication among Logical Processes. The algorithm in [63] also allows normal processing to continue during the Global Virtual Time determination phase. Global Virtual Time is required only for the purpose of throttling forward prediction in Virtual Network Configuration, that is, it governs how far into the future the system predicts. An alternative method for throttling the Virtual Network Configuration system makes use of the accurate time provided by the Global Positioning System receiver and is discussed later in the analysis and implementation chapters.

Virtual Network Configuration messages contain the Send Time (TS), Receive Time (TR), Anti-toggle (A) and the actual message value itself (M). The TR is the time this message is predicted be valid at the destination Logical Process. It is not the link transfer time from source to destination Logical Process. The TS is the time this message was sent by the originating Logical Process. The “A” field is the anti-toggle and is used for creating an anti-message to remove the effects of false messages as described later. A message also contains a field for current Real Time (RT). This is used to differentiate a real message from a virtual message. A message which is generated and time-stamped with the current time is called a real message. Messages which contain future event information and are time-stamped with a time greater than current time are called virtual messages. If a message arrives at an Logical Process out of order or with invalid information, it is called a false message. A false message will cause an Logical Process to rollback. The Logical Process structures and message fields are shown in Table 3.2, Table 3.3 and in Figure 3.4.

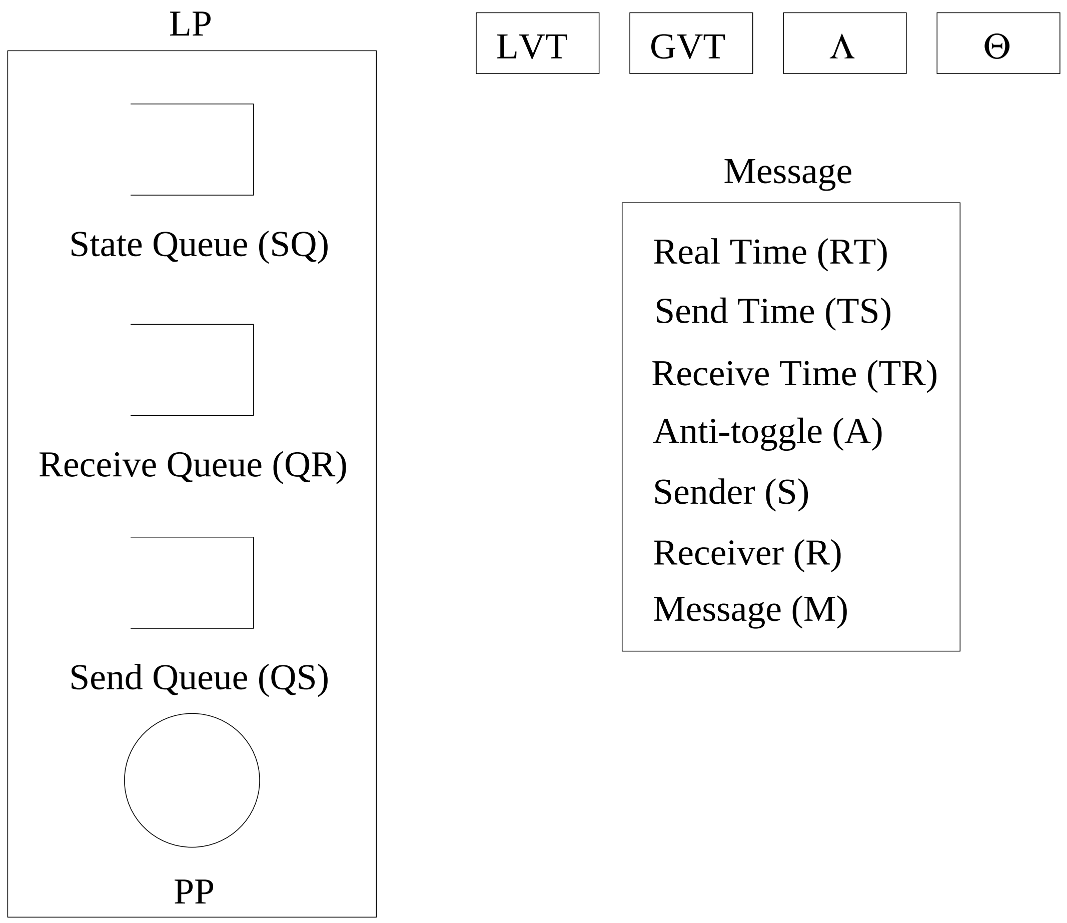


Figure 3.4: The VNC Logical Process and Message.

The Virtual Network Configuration algorithm requires a driving process to predict future events and inject them into the system. The driving process acts a source of virtual messages for the Virtual Network Configuration system. All other processes react to virtual messages. For example, the Global Positioning System receiver process runs in real-time providing current time and location information and has been modified to inject future predicted time and location messages as well.

A rollback is triggered either by messages arriving out of order at the Receive Queue of an Logical Process or by a predicted value previously computed by this Logical Process which is beyond the allowable tolerance. In either case, rollback is a mechanism by which an Logical Process returns to a known correct state. The rollback occurs in three phases. In the first phase, the Logical Process state is restored to a time strictly earlier than the Receive Time of the false message. In the second phase, anti-messages are sent to cancel the effects of any invalid messages which had been generated before the arrival of the false message. An anti-message contains exactly the same contents as the original message with the exception of an anti-toggle bit which is set. When the anti-message and original message meet, they are both annihilated. The final phase consists of executing the Logical Process forward in time from its rollback state to the time the false message arrived. No messages are canceled or sent between the time to which the Logical Process rolled back and the time of the false message. Because these messages are correct, there is no need to cancel or re-send them. This increases performance, and it prevents additional rollbacks. Note that another false message or anti-message may arrive before this final phase has completed without causing problems. The Virtual Network Configuration Logical Process has the contents shown in Table 3.2, the message fields are shown in Table 3.3, and the message types are listed in Table 3.4 where t is the real time at the receiving Logical Process.

3.2.1 Multiple Future Event Architecture

It is possible to anticipate alternative future events using a direct extension of the basic Virtual Network Configuration algorithm [113]. The driving process generates multiple virtual messages, one for each possible future event with corresponding probabilities

Structure	**Description**
Receive Queue (QR)	Ordered by message receive time (TR)
Send Queue (QS)	Ordered by message send time (TS)
Local Virtual Time (LVT)	$LVT = \inf RQ$
Current State (CS)	State of the logical and physical process
State Queue (SQ)	States (CS) are periodically saved
Sliding Lookahead Window (Λ)	$SLW = (t, t + \Lambda]$
Tolerance (Θ)	Allowable deviation

Table 3.2: Virtual Network Configuration Logical Process Structures.

Field	**Description**
Send Time (TS)	LVT of sending process when message is sent
Receive Time (TR)	Scheduled time to be received by receiving process
Anti-toggle (A)	Identifies message as normal or antimessage
Message (M)	The actual contents of the message
Real Time (RT)	The GPS time at which the message originated

Table 3.3: Virtual Network Configuration Message Fields.

Virtual Message	$RT > t$
Real Message	$RT \leq t$

Table 3.4: Virtual Network Configuration Message Types.

of occurrence, or a ranking, for each event. Instead of a single Receive Queue for each Logical Process, multiple Receive Queues for each version of an event are created dynamically for each LP. The logical process can dynamically create Receive Queues for each event and give priority to processing messages from the most likely versions' Receive Queues. This enhancement to Virtual Network Configuration has not been implemented. This architecture for implementing alternative futures, while a simple and natural extension of the Virtual Network Configuration algorithm, creates additional messages and increases the message sizes. Messages require an additional field to identify the probability of occurrence and an event identifier.

3.2.2 Example of Virtual Network Configuration Operation

A specific simple example of the Virtual Network Configuration algorithm is shown in Figures 3.5 through 3.9. The description of the algorithm begins with a system which has just been powered up and is generating real messages. The driving process will begin to predict future events and inject virtual messages based on those events. An Logical Process receives a message and must first determine whether it is virtual or real by examining the RT field. If the message is a virtual message, the Logical Process compares the message with its Local Virtual Time to determine whether a rollback is necessary due to an out of order message. If the message has not arrived in the past relative to the Logical Process's Local Virtual Time, the message then enters the Receive Queue in order by Receive Time. The Logical Process takes the next message from the Receive Queue, updates its Local Virtual Time, and processes the message. If an outgoing message is generated, a copy of the message is saved in the Send Queue, the Receive Time is set, and the Send Time is set to the current Local Virtual Time. The message is then sent to the destination Logical Process. If the virtual message arrived out of order, the Logical Process must rollback as described previously. Figure 3.5 is an example of an Edge Switch Logical Process which determines handoff. Input messages arrive from Logical Processes on the Remote Node and enter Logical Processes on the Edge Switch which cause messages to arrive at the Handoff calculation Logical Process shown in Figure 3.5. At the start of the Logical Process's execution, LVT and GVT are set to zero, Lookahead (Λ) is set to two minutes and a tolerance (Θ) of 2 units of distance is allowed between predicted and actual location values for this process.

0	LVT	0	GVT	120	Λ	2	Θ

Receive Queue (QR)

Real Time (RT)	0
Send Time (TR)	10
Receive Time (RT)	15
Anti-toggle (A)	0
Sender (S)	KBOWTN
Receiver (R)	KBOWTA
Message (M)	5,5

State

State Queue (SQ)

ES Positions	5,5
RN Positions	10,10
Time	0

Send Queue (QS)

Empty	Empty

Figure 3.5: Edge Switch Handoff Logical Process: First Virtual Message Arrives.

In Figure 3.6, an output message is sent indicating to the Remote Node that it should associate with this Edge Switch. Also, the next virtual message has arrived, so the LVT will be set to the Receive Time of the virtual message. The new Logical Process state is saved in the State Queue.

The first real message arrives in Figure 3.7 with an Real Time value of 5. The position in the message value (5,6) is compared with the value (5,5) which is the position

15	LVT	0	GVT	120	Λ	2	Θ

Receive Queue (QR)

Message

Real Time (RT)	2	~~0~~
Send Time (TR)	15	~~10~~
Receive Time (RT)	30	~~15~~
Anti-toggle (A)	0	~~0~~
Sender (S)	KBOWTN	~~KBOWTN~~
Receiver (R)	KBOWTA	~~KBOWTA~~
Message (M)	6,7	~~5,5~~

State

State Queue (SQ)

ES Positions	6,7	5,5
RN Positions	10,10	10,10
Time	15	0

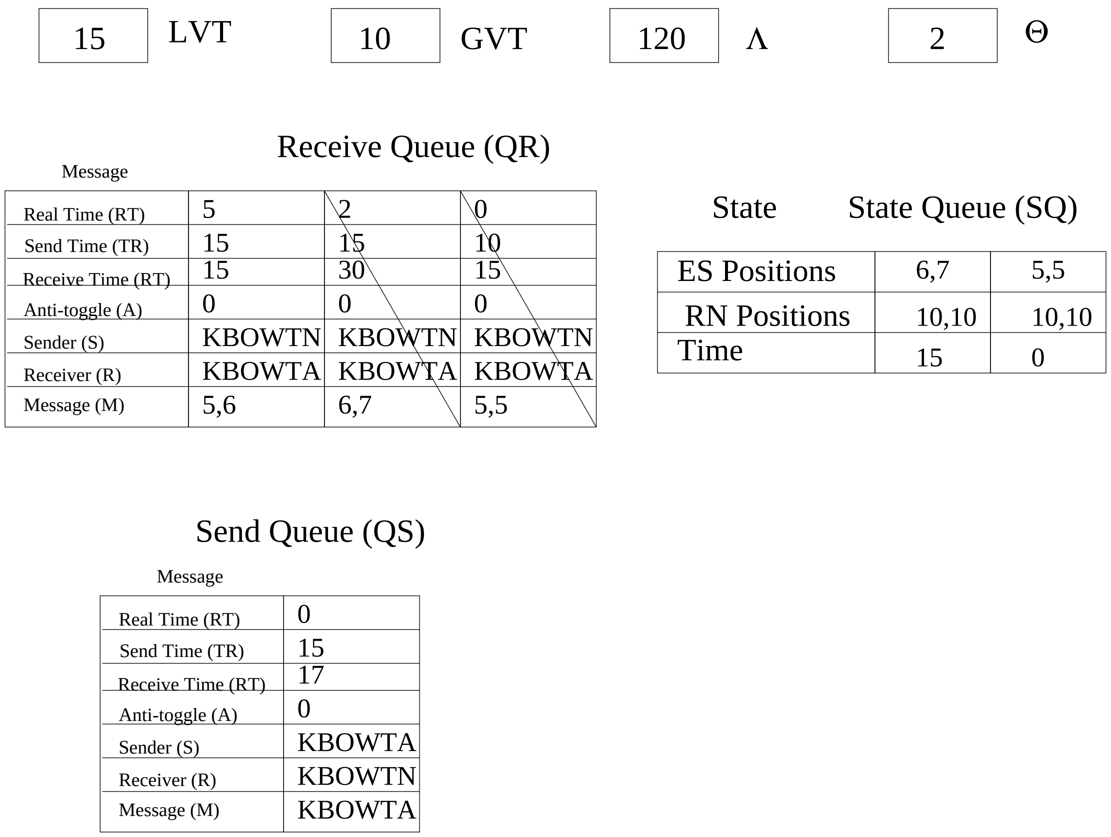
Message

Send Queue (QS)

Real Time (RT)	0
Send Time (TR)	15
Receive Time (RT)	17
Anti-toggle (A)	0
Sender (S)	KBOWTA
Receiver (R)	KBOWTN
Message (M)	KBOWTA

Figure 3.6: Edge Switch Handoff Logical Process: Output Message Sent.

in the State Queue with the closest time at which the state was saved. It is found to be within the required tolerance, thus, no rollback occurs and no new messages need to be sent. Although the Global Virtual Time is not necessary for Virtual Network Configuration operation as discussed in the next chapter, it is included in this example. The Global Virtual Time is updated in Figure 3.7 to indicate the time that the entire system has predicted into the future.



15	LVT	10	GVT	120	Λ	2	Θ

Receive Queue (QR)

Message			
Real Time (RT)	5	~~2~~	~~0~~
Send Time (TR)	15	~~15~~	~~10~~
Receive Time (RT)	15	~~30~~	~~15~~
Anti-toggle (A)	0	~~0~~	~~0~~
Sender (S)	KBOWTN	~~KBOWTN~~	~~KBOWTN~~
Receiver (R)	KBOWTA	~~KBOWTA~~	~~KBOWTA~~
Message (M)	5,6	~~6,7~~	~~5,5~~

State State Queue (SQ)

ES Positions	6,7	5,5
RN Positions	10,10	10,10
Time	15	0

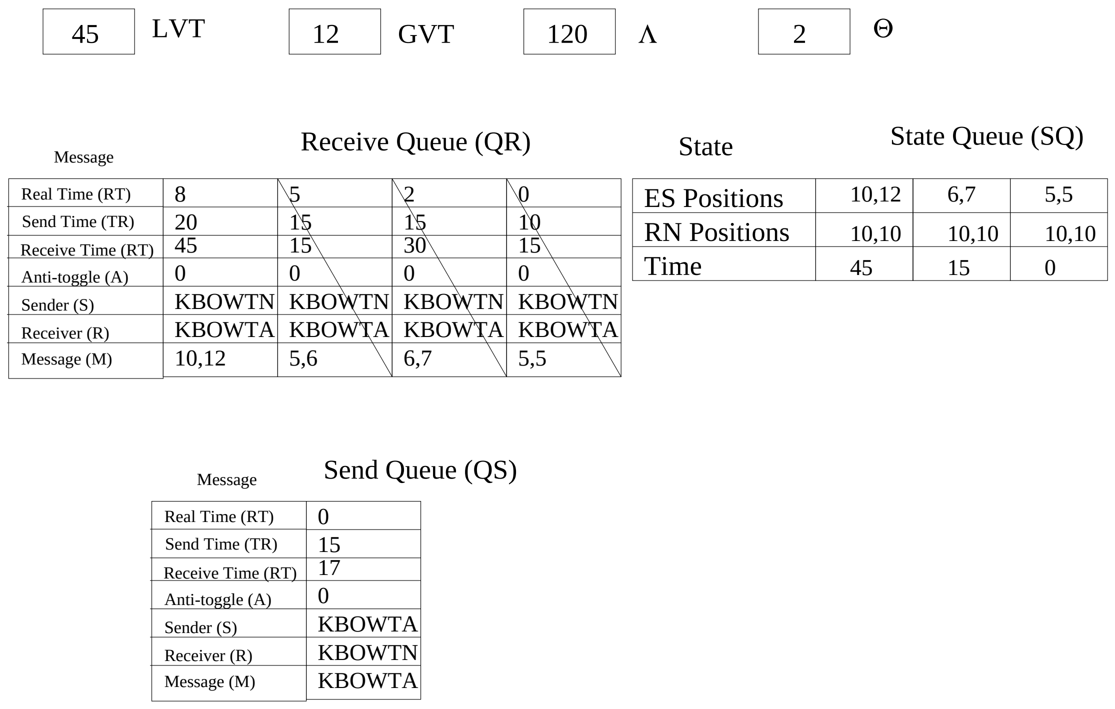
Send Queue (QS)

Message	
Real Time (RT)	0
Send Time (TR)	15
Receive Time (RT)	17
Anti-toggle (A)	0
Sender (S)	KBOWTA
Receiver (R)	KBOWTN
Message (M)	KBOWTA

Figure 3.7: Edge Switch Handoff Logical Process: First Real Message Arrives.

Another virtual message arrives in Figure 3.8. The message has arrived in proper order, therefore, the Local Virtual Time is updated, the message is processed and the state is saved. No output message was generated since no new Edge Switch-Remote Node association was determined by the Physical Process for this Logical Process.

An example of out of tolerance rollback is illustrated in Figure 3.9. A real mes-



45	LVT	12	GVT	120	Λ	2	Θ

Receive Queue (QR)

Message				
Real Time (RT)	8	~~5~~	~~2~~	~~0~~
Send Time (TR)	20	~~15~~	~~15~~	~~10~~
Receive Time (RT)	45	~~15~~	~~30~~	~~15~~
Anti-toggle (A)	0	~~0~~	~~0~~	~~0~~
Sender (S)	KBOWTN	~~KBOWTN~~	~~KBOWTN~~	~~KBOWTN~~
Receiver (R)	KBOWTA	~~KBOWTA~~	~~KBOWTA~~	~~KBOWTA~~
Message (M)	10,12	~~5,6~~	~~6,7~~	~~5,5~~

State Queue (SQ)

State			
ES Positions	10,12	6,7	5,5
RN Positions	10,10	10,10	10,10
Time	45	15	0

Send Queue (QS)

Message	
Real Time (RT)	0
Send Time (TR)	15
Receive Time (RT)	17
Anti-toggle (A)	0
Sender (S)	KBOWTA
Receiver (R)	KBOWTN
Message (M)	KBOWTA

Figure 3.8: Edge Switch Handoff Logical Process: Second Virtual Message Arrives.

sage arrives and its message contents are compared with the closest saved state value. Because the message value is found to be out of tolerance, all state queue values with times greater than the receive time of the real message are discarded. The send queue message anti-toggle is set and the anti-message is sent. The discarded states and messages are slashed in Figure 3.9. The rollback causes the Logical Process to go back to time 15 because that is the time of the most recent saved state which is less than the time of the out-of-tolerance message's Receive Time.

After the rollback occurs, the Logical Process has the state shown in Figure 3.10. The Local Virtual Time has been adjusted to time 15 and the Logical Process continues normal processing. The algorithm is presented in more detail in the next section.

45	LVT	12	GVT	120	Λ	2	Θ

Receive Queue (QR)

Message					
Real Time (RT)	15	~~8~~	~~5~~	~~2~~	~~0~~
Send Time (TR)	20	~~20~~	~~15~~	~~15~~	~~10~~
Receive Time (RT)	45	~~45~~	~~15~~	~~30~~	~~15~~
Anti-toggle (A)	0	~~0~~	~~0~~	~~0~~	~~0~~
Sender (S)	KBOWTN	~~KBOWTN~~	~~KBOWTN~~	~~KBOWTN~~	~~KBOWTN~~
Receiver (R)	KBOWTA	~~KBOWTA~~	~~KBOWTA~~	~~KBOWTA~~	~~KBOWTA~~
Message (M)	7,8	~~10,12~~	~~5,6~~	~~6,7~~	~~5,5~~

State Queue (SQ)

State			
ES Positions	~~10,12~~	6,7	5,5
RN Positions	~~10,10~~	10,10	10,10
Time	~~45~~	15	0

Send Queue (QS)

Message	
Real Time (RT)	~~0~~
Send Time (TR)	~~15~~
Receive Time (RT)	~~17~~
Anti-toggle (A)	~~1~~
Sender (S)	~~KBOWTA~~
Receiver (R)	~~KBOWTN~~
Message (M)	~~KBOWTA~~

Figure 3.9: Edge Switch Handoff Logical Process: Rollback Occurs.

15	LVT	12	GVT	120	Λ	2	Θ

Receive Queue (QR)

Message					
Real Time (RT)	~~15~~	~~8~~	~~5~~	~~2~~	~~0~~
Send Time (TR)	~~20~~	~~20~~	~~15~~	~~15~~	~~10~~
Receive Time (RT)	~~45~~	~~45~~	~~15~~	~~30~~	~~15~~
Anti-toggle (A)	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
Sender (S)	~~KBOWTN~~	~~KBOWTN~~	~~KBOWTN~~	~~KBOWTN~~	~~KBOWTN~~
Receiver (R)	~~KBOWTA~~	~~KBOWTA~~	~~KBOWTA~~	~~KBOWTA~~	~~KBOWTA~~
Message (M)	~~7,8~~	~~10,12~~	~~5,6~~	~~6,7~~	~~5,5~~

Send Queue (QS)

Empty

State Queue (SQ)

State			
ES Positions	~~10,12~~	6,7	5,5
RN Positions	~~10,10~~	10,10	10,10
Time	~~45~~	15	0

Figure 3.10: Edge Switch Handoff Logical Process: After Rollback Occurs.

3.3 Pseudocode Specification for Virtual Network Configuration

The Virtual Network Configuration algorithm requires both Driving Processes and Logical Processes. Driving Processes predict events and inject virtual messages into the system. Logical Processs react to both real and virtual messages. The Virtual Network Configuration Algorithm for a driving process is shown in Algorithm 1. The operation of the both the driving process and the logical process repeat indefinitely. If the Driving Process has not exceeded its lookahead time on line 2, a new value Δ time units into the future is computed by the function $C(t)$ and the result is assigned to the message (M) and sent on line 5. The receive time, which is the time at which this message value is to be valid is assigned to (M) in line 4. Every $UpdatePeriod$ time units, real messages are generated as shown in lines 7 and 8.

Algorithm 1: VNC Driving Process Algorithm.

```
VncDriver($UpdatePeriod$, $\Lambda$)
(1)   repeat
(2)     if $GVT \leqslant t + \Lambda$
(3)       $M.val \leftarrow C(LVT + \Delta)$
(4)       $M.rt \leftarrow LVT + \Delta$
(5)       Send(M)
(6)     if $t \mod UpdatePeriod = 0$
(7)       $M.val \leftarrow GPS.pos$
(8)       Send(M)
```

The Virtual Network Configuration Algorithm for an Logical Process is specified in Algorithm 2. On line 4, $\inf$ is infimum. On line 6, the next message from the Receive Queue is checked to determine whether the message is real. If the message is real, line 6 retrieves the state that was saved closest to the receive time of the message and checks whether the values of the saved state are within tolerance. If the tolerance is exceeded, the process will rollback. Also, if the message is received in the past relative to this process's LVT, the process will rollback as shown in line 7. In line 8, the pre-computed and cached value in the state queue is committed. Committing a value is an irreversible action because it cannot be rolled back once committed. If the process's LVT has not exceeded its lookahead time as determined in line 9, then the virtual message is processed in lines 10 through 15. The function $C_1(M, LVT)$ in line 10 represents the computation of the new state. The function $C_1(M, LVT)$ returns the state value for this Logical Process and updates the LVT to the time at which that value is valid. The function $C_2(M, LVT)$ in line 12 represents the computation of a new message value.

Algorithm 2: VNC Logical Process Algorithm.

```
VNC(Θ, Λ)
(1)   $LVT \leftarrow 0$
(2)   $t \leftarrow GPS.t$
(3)   **repeat**
(4)       $M \leftarrow \inf M.tr \in QR$
(5)       $CS(t).val \leftarrow C_1(M, t)$
(6)       **if** $(M.rt \leqslant t) \wedge (\mid SQ(t).val - \Theta \mid > CS(t).val)$ **then** Roll-
          back()
(7)       **if** $M.rt < LVT$ **then** Rollback()
(8)       **if** $M.rt \leqslant t$ **then** Commit($SQ : SQ.t =\approx M.rt$)
(9)       **if** $LVT + \Lambda \leqslant GVT$
(10)          $SQ.val \leftarrow C_1(M, LVT)$
(11)          $SQ.t \leftarrow LVT$
(12)          $M.val \leftarrow C_2(M, LVT)$
(13)          $M.rt \leftarrow LVT$
(14)          $QS \leftarrow M$
(15)          Send(M)
```

Chapter 4

Algorithm Analysis

This goal of this chapter is to analyze the performance of Virtual Network Configuration. The characteristics of Virtual Network Configuration to be analyzed are speedup, potentially wasted resources, and bandwidth overhead. Speedup is the ratio of the time required to perform an operation without Virtual Network Configuration divided by the time required with Virtual Network Configuration. Wasted resources are defined to be resources which are temporarily allocated but never used due to prediction inaccuracy. Bandwidth overhead is the ratio of the amount of additional bandwidth required by a Virtual Network Configuration system divided by the amount of bandwidth required by a none Virtual Network Configuration system.

Because the Logical Processs of a Virtual Network Configuration system are asynchronous, the Logical Processs can take maximum advantage of parallelism. However, messages among processes may arrive at the destination process out-of-order. A Petri-Net model is used to quantify the amount of messages that arrive out-of-order for a particular Virtual Network Configuration system. Petri-Nets are commonly used for synchronization analysis where "places", usually shown as circles, represent entities such as producers, consumers, or buffers, and "transitions", shown as squares allow "tokens" shown as dots, to move from one place to another. In this analysis, Petri-Net tokens represent Virtual Network Configuration messages and Petri-Net places represent Virtual Network Configuration Logical Processs. Characteristics of Petri-Nets are used to determine the likelihood of out-of-order messages. The likelihood of the occurrence of out-of-order messages and out-of-tolerance messages is required by an equation that is developed in this chapter to describe the speedup of Virtual Network Configuration. After analyzing the speedup, the prediction accuracy and bandwidth are analyzed. The chapter concludes by considering enhancements and optimizations such as implementing multiple future events, eliminating the Global Virtual Time calculation, and elimination of real messages when they are not required.

4.1 Utility Function Analysis

Performance analysis of the Virtual Network Configuration (VNC) algorithm must take into account accuracy as well as speed. An inaccurate configuration can result in committed resources which are never used and thus wasted, or in not committing enough resources when needed thus causing a delay. Unused resource allocation must be minimized. Many of the mobile wireless Asynchronous Transfer Mode mechanisms previously mentioned depend on keeping resources permanently allocated, such as Asynchronous Transfer Mode Virtual Circuits in [1]. Virtual Network Configuration does not require permanent over-allocation of resources, however, the Virtual Network Configuration algorithm may make a false prediction which *temporarily* establishes resources which may never be used. A Virtual Network Configuration system whose tolerances are reduced in order to produce more accurate results will have fewer unused allocated resources, however, the tradeoff is a reduction in speedup.

Equation 4.1 quantifies the advantage of using Virtual Network Configuration where

$$U_{VNC} = \eta\Phi_s - \alpha\Phi_w - \beta\Phi_b \tag{4.1}$$

η is the expected speedup using Virtual Network Configuration over a non-VNC system, Φ_s is the marginal utility function of the configuration speed, and α is the expected quantity of wasted resources other than bandwidth, and Φ_w is the marginal utility function of the allocated but unused resource. An example of a resource which may be temporarily wasted due to prediction error is a Virtual Circuit (VC) which may be established temporarily and not used. The expected bandwidth overhead is represented by β and Φ_b is the marginal utility function of bandwidth.

The marginal utility functions Φ_s, Φ_w and Φ_b are subjective functions which describe the value of a particular service to the user. The functions Φ_s, Φ_w and Φ_b may be determined by monetary considerations and user perceptions. The following sections develop propositions which describe the behavior of the Virtual Network Configuration algorithm and from these propositions equations for η, α and β are defined.

4.2 Petri-Net Analysis for Virtual Network Configuration

This goal of this section is to determine the probability of out-of-order messages arriving at an Logical Process and to determine the expected proportion of out-of-order messages ($E[X]$) and the probability of rollback due to out-of-order messages (P_{oo}). Another goal of this section is to develop a new and simpler approach to analyzing Time Warp based algorithms in general and Virtual Network Configuration in particular. The contribution of this section is unique because most current optimistic synchronization analysis has been explicitly time-based yielding limited results except for very specific cases. The approach taken in this section is topological; timing is implicit rather than explicit. A Condition Event Network is used in this analysis because it is the simplest form of a Petri Net that is ideal for studying Virtual Network Configuration synchronization behavior.

A Condition Event Network network consists of condition and transition elements that contain tokens. Tokens reside in condition elements. When all condition elements leading to a transition element contain a token, several changes take place in the Condition Event Network network. First, the tokens are removed from the conditions that triggered the event, the event occurs, and finally tokens are placed in all condition outputs from the transition that was triggered. Multiple tokens in a condition and the uniqueness of the tokens is irrelevant in a Condition Event Network Net. In this analysis, tokens represent virtual messages, conditions represent processes, and transitions represent Logical Process interconnections. The notation from [92] is used: $\Sigma = (B, E;\ F, C)$ is a Condition Event Network Net where B is the set of conditions, E is the set of transitions, and $F \subseteq (B \times E) \cup (E \times B)$ where $\cup$ is union and $\times$ is the cross product of all conditions and transitions. A marking is the set of conditions containing tokens at any given time during Condition Event Network operation and C is the set of all possible sets of markings of Σ. The input conditions to a transition will be written as "pre-e" and the output conditions will be written as "post-e". Let $c \subseteq C$, then a transition $e \in E$ is triggered when pre-$e \subseteq (c \subseteq B)$ and post-$e \cap c = \emptyset$. If c is the current set of enabled conditions and after the next transition (e) the new set of enabled conditions is c', then this is represented more compactly as $c[e\rangle c'$. Condition Event Network networks provide insight into liveness, isomorphism, reachability, a method for determining synchronous behavior, and behavior based on the topology of Virtual Network Configuration (VNC) Logical Process (LP) communication. An example of a Condition Event Network Net is shown in Figure 4.1 for an Edge Switch. The initial marking begins with a token in the Initialize condition. Receipt of a **MYCALL** packet causes the transmission of a **NEWSWITCH** message event to occur. After a Timeout occurs, the Edge Switch begins to handle Remote Node activity. Clearly, the Condition Event Network network shown in Figure 4.1 is a subset of the Finite State Machine (FSM) from Table 2.3. Every Finite State Machine has an equivalent Condition Event Network Net [87, p. 42].

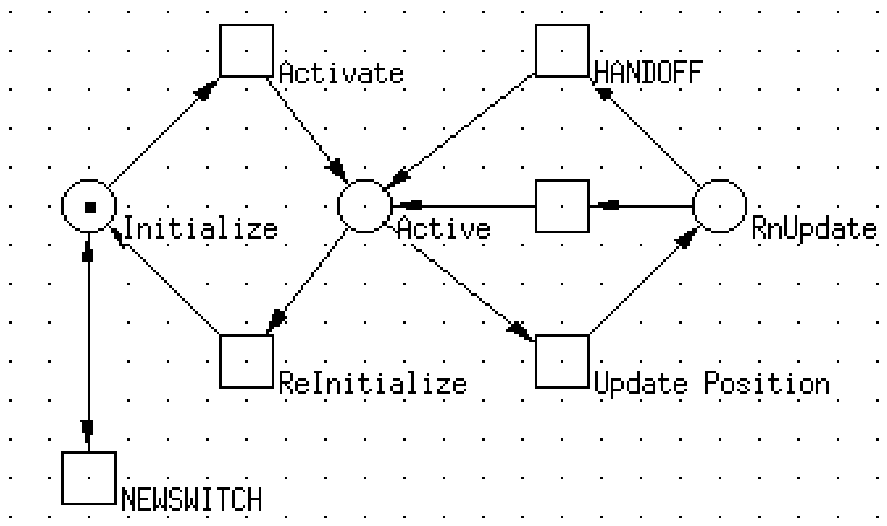


Figure 4.1: Subset of NCP FSM as a C/E Net.

Some common terminology and concepts are defined next that will be needed for a topological analysis of Virtual Network Configuration. These terms and concepts are introduced in a brief manner and build upon one another. Their relationship with Virtual Network Configuration will soon be made clear. The following notation is used: "$\neg$ " means "logical not", "$\exists$" means "there exists", "$\forall$" means "for each", "$\wedge$" means "logical and", "$\vee$" means "logical or", "$\in$" means that an element is a member of a set, "$\equiv$" means "defined as", and "$\rightarrow$" defines a mapping or function. Also, $a \prec b$ indicates an ordering between two elements, a and b, such that a precedes b in some relation.

"⇒" means "logical implication" and "↔" means "logical equivalence".

A region of a particular similarity relation $(\cdot)$ of $B \subseteq A$ means that $\forall\, a, b \in B : a \cdot b$ and $\forall\, a \in A : a \notin B \Rightarrow \exists\, b \in B : \neg\,(a \cdot b)$. This means that the relation is "full" on B and B is a maximal subset on that the relation is full. In other words, a graph of the relation $(\cdot)$ would show B as the largest fully connected subset of nodes in A.

Let " $\underline{\text{li}}$ " represent a linear ordering such that $a\ \underline{\text{li}}\ b \leftrightarrow (a \prec b) \vee (b \prec a) \vee (a \equiv b)$. Let " $\underline{\text{co}}$ " represent a concurrent ordering $a\ \underline{\text{co}}\ b \leftrightarrow \neg\,(a\ \underline{\text{li}}\ b) \vee (a \equiv b)$. Figure 4.2 illustrates a region of $\underline{\text{co}}$ that contains $\{a, c\}$ and region of $\underline{\text{li}}$ that contains $\{a, b, d\}$ where $\{a, b, c, d\}$ represents Logical Processs and the relation is "sends a message to". Trivially, if every process in the Virtual Network Configuration system is a region of $\underline{\text{li}}$ then regardless of how many driving processes there are, no synchronization is necessary since there exist no concurrent processes.

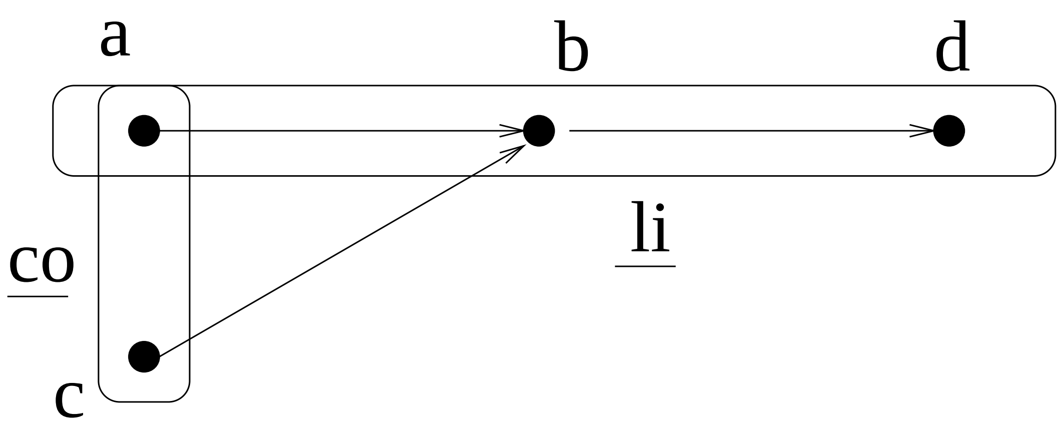


Figure 4.2: Demonstration of $\underline{\text{li}}$ and $\underline{\text{co}}$.

Let D is the set of driving processes and R be the set of the remaining processes in the Virtual Network Configuration system. Then $D \prec R \leftrightarrow \forall\, d \in D\ \forall\, r \in R : (d \prec r) \vee (d\ \underline{\text{co}}\ r)$. In order for the virtual messages that originate from D to be used, $D \prec R$ where R are the remaining non-driving processes. The relation used here is again assumed to be "sends a message to".

In the remaining definitions, let A, B, and C be arbitrary sets where $B \subseteq A$ used for defining additional operators. Let $B \preceq C \equiv \forall\, b \in B \,\forall\, c \in C : b \prec c \vee b \;\underline{\text{co}}\; c$. Let $B^- \equiv \{a \in A \mid \{a\} \preceq B\}$ and $B^+ \equiv \{a \in A \mid B \preceq \{a\}\}$ where $\mid$ means "such that". Also, let $\overline{B} \equiv \{b \in B \mid \forall\, b' \in B : (b \;\underline{\text{co}}\; b') \vee (b \prec b')\}$ and $\underline{B} \equiv \{b \in B \mid \forall\, b' \in B : (b \;\underline{\text{co}}\; b') \vee (b' \prec b)\}$. This is illustrated in Figure 4.3 where all nodes are in the set A and B is the set of nodes that lie within the circle. B^- is the set $\{a, b, c, d, f\}$ and $\overline{B}$ is the set $\{b\}$.

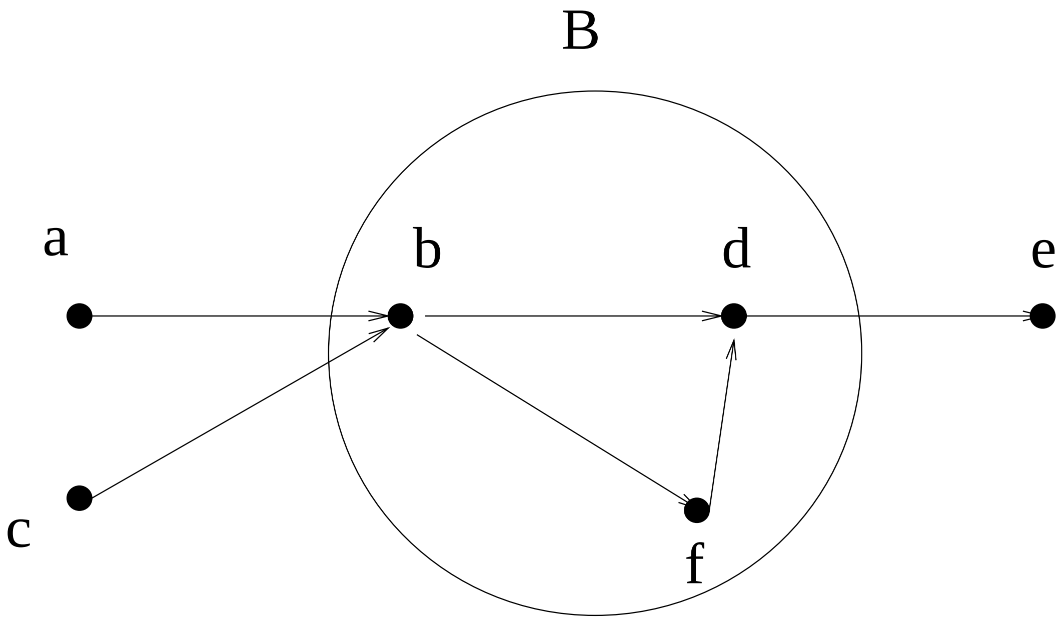


Figure 4.3: Illustration of B^- and $\overline{B}$.

An occurrence network (K) is a Condition Event Network network that is related to the operation of a particular Condition Event Network network (Σ). The occurrence network (K) begins as an empty Condition Event Network network; conditions and events are added to K as Σ operates. Thus, K represents a particular sample of operation of Σ. There can be multiple events in Σ which are capable of firing, but only one event is chosen to fire, thus it is possible that a particular Σ will not always generate the same occurrence net (K) each time it operates. Note that K has some special properties. The condition elements of K have one and only one transition, because only one token in Σ may fire from a given condition. Also, K is cycle free because K represents the operation of Σ.

A few more definitions are required before the relation described above between K and Σ can be formally defined. This relationship is called a Petri-Net process. Once the Petri-Net process is defined, a measure for the "out-of-orderness" of messages can be developed based on synchronic distance. A *line* is a subset that is a region of $\underline{\text{li}}$ and a *cut* is a subset that is a region of $\underline{\text{co}}$. A slice (" $\underline{\text{sl}}$ ") is a cut of an occurrence network (K) containing condition elements and $\underline{\text{sl}}\,(K)$ is the set of *all* slices of K. The region of $\underline{\text{co}}$ shown in Figure 4.2 illustrates a cut where nodes represent conditions and the relation defines an event from one condition to another in a Condition Event Network Network.

A formal definition of the relation between an occurrence net and a Condition Event Network net is given by a Petri-Net process. A Petri-Net process (p) is defined as a mapping from a network K to a Condition Event Network Network Σ, $p : K \rightarrow \Sigma$, such that each slice of K is mapped injectively (one-to-one) into a marking and $(p(\text{pre-}t) = \text{pre-}p(t)) \wedge (p(\text{post-}t) = \text{post-}p(t))$. Also note that p^{-1} is used to indicate the inverse mapping of p. Think of K as a particular sample of the operation of a Condition Event Network Network. A Condition Event Network Network can generate multiple processes Another useful characteristic is whether a network is K-dense. A network is K-dense if and only if every $\underline{\text{sl}}\,(K)$ has a non-empty intersection with every region of $\underline{\text{li}}$ in K. This means that each slice intersects every sequential path of operation.

All of the preceding definitions have been leading towards the development of a measure for the "out-of-orderness" of messages that is relatively simple to calculate because it does not rely on explicit time values or distributions. In the following explanation, a measure is developed for the synchronization between events. Consider D_1 and D_2 that are two slices of K and M is a set of events in a Condition Event Network Network.

$\mu(M, D_1, D_2)$ is defined as $\mid M \cap D_2^+ \cap D_1^- \mid - \mid M \cap D_1^- \cap D_2^+ \mid$. Note that $\mu(M, D_1, D_2) = -\mu(M, D_2, D_1)$. Thus $\mu(M, D_1, D_2)$ is a number that defines the number of events between two specific slices of a net.

Let $(p : K \rightarrow \Sigma) \in \pi_\Sigma$ where π_Σ is the set of all finite processes of Σ. A term known as "variance" is defined that describes the number of events across all slices of a net (K). The variance of T_Σ is $\nu(p, T_1, T_2) \equiv \max\{\mu(p^{-1}(T_1), D_1, D_2) - \mu(p^{-1}(T_2), D_1, D_2) \mid D_1, D_2 \in \underline{\mathrm{sl}}\,(K)\}$. Also, note that $\nu(p, T_1, T_2) = \nu(p, T_2, T_1)$ where and $T_1, T_2 \subseteq T_\Sigma$. This defines a measure of the number of events across all slices of a net (K).

The synchronic distance ($\sigma(T_1, T_2) = \sup\{\nu(p, T_1, T_2) \mid p \in \pi_\Sigma\}$) is the supremum of the variance in all finite processes. This defines the measure of "out-of-orderness" across all possible K. By determining the synchronic distance, a measure for the likelihood of rollback in Virtual Network Configuration can be defined that is dependent on the topology and is **independent of time**. Further details on synchronic distance and the relation of synchronic distance to other measures of synchrony can be found in [117]. A more intuitive method for calculating the synchronic distance is to insert a virtual condition into the Condition Event Network net. This condition has no meaning or effect on Condition Event Network operation. It is allowed to hold multiple tokens and begins with enough tokens so that it can emit a token whenever a condition connected to its output transition is ready to fire. The virtual condition has inputs from all members of T_1 and output transitions of all members of T_2. The synchronic distance is the maximum variation in the number of tokens in the virtual condition. The greater the possibility of rollback, the larger the value of $\sigma(T_1, T_2)$. A simple example in Figure 4.4 intuitively illustrates what the synchronic distance means. Using the virtual condition method to calculate the synchronic distance between $\{a, b\}$ and $\{c, d\}$ in the upper Condition Event Network Network, the synchronic distance is found to be two. By adding two more conditions and another transition to the Condition Event Network network, the synchronic distance of the lower Condition Event Network Network shown in Figure 4.4 is one. The larger the value of $\sigma(T_1, T_2)$ the less synchronized the events in sets T_1 and T_2. If these events indicate message transmission, then the less synchronized the events, the greater the likelihood that the messages based on events T_1 and T_2 will be out-of-order. This allows the likelihood of out-of-order message arrival at an Logical Process to be determined based on the inherent synchronization of a system. However, a completely synchronized system will not gain the full potential provided by optimistic parallel synchronization.

A P/T Network is similar to a Condition Event Network network except that a P/T Net allows multiple tokens in a place and multiple tokens may be required to cause a transition to fire. Places are defined by the set S and transitions by the set T. The operation of a P/T network can be described by a matrix. The rows of the matrix represent places and the columns represent transitions. The last column of the matrix represents the current number of tokens in a place. Each element of the matrix contains the number of tokens which either leave (negative integer) or enter (positive integer) a place when the transition fires. When a transition fires, the column corresponding to the transition is added to the last column of the matrix. Thus, the last column of the matrix changes as the number of nodes in each place change. The matrix representation of a P/T Network is shown in Matrix 4.2, where $LP_n \in S$, $c_n \in T$ and $w_{i,j}$ is the weight or number of tokens required by link j to fire or the number of tokens generated by place i. Note that LP_n and c_n bordering Matrix 4.2 indicate labels for rows and columns. Note also that there exists a duality between places and transitions such that places and

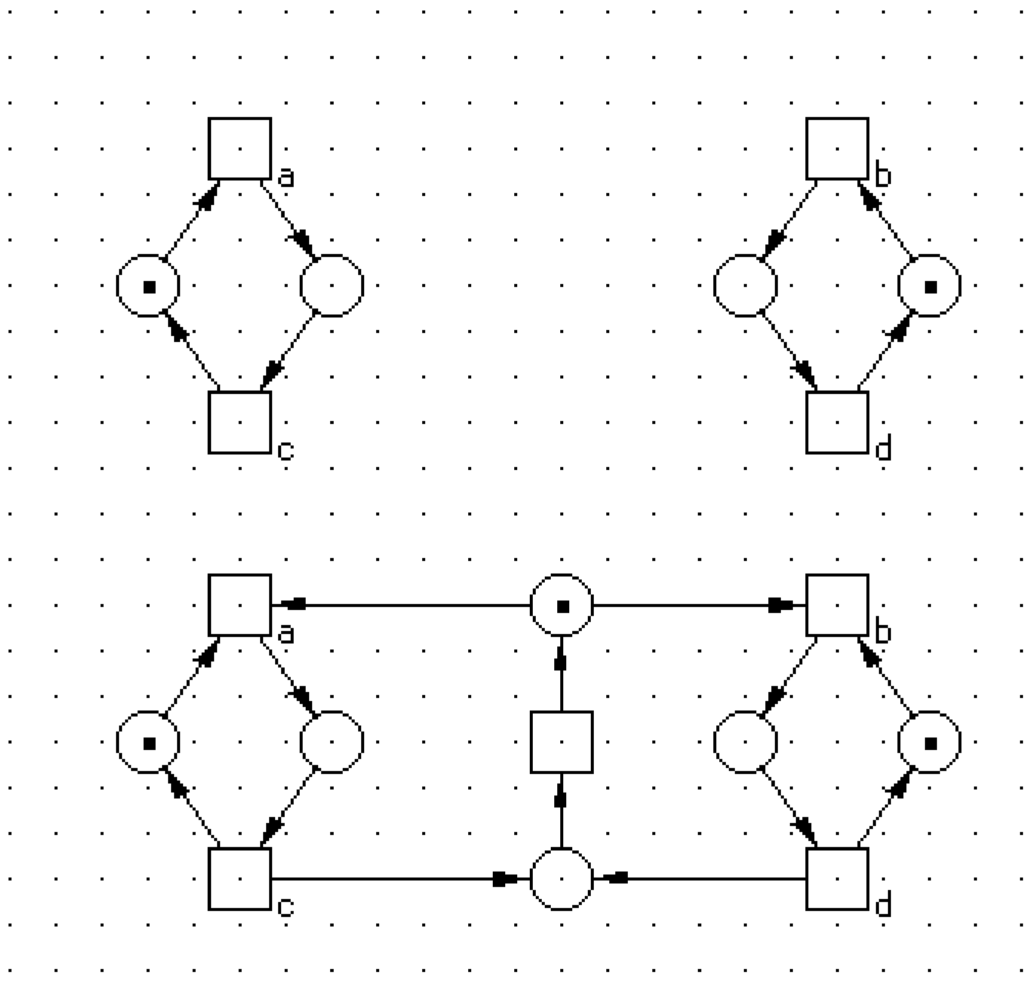


Figure 4.4: Example of Synchronic Distance.

transitions can be interchanged [87, p. 13]. P/T networks can be extended from the state representation of Condition Event Network networks to examine problems involving quantities of elements in a system, such as producer/consumer problems. The places in this analysis are analogous to Logical Processs because they produce and consume both real and virtual messages. Transitions in this analysis are analogous to connections between Logical Processs, and tokens to messages. The weight, or number of tokens, is $-w_{i,j}$ for outgoing tokens and $w_{i,j}$ for incoming tokens. The current marking, or expected value of the number of tokens held in each place is given in column vector $\vec{m_N}$. A transition to the next state is determined by $\vec{m_{N+1}} = \vec{m_N} + \vec{c_i}$ where $\vec{c_i}$ is the column vector of the transition that fired and N is the current matrix index.

$$M_N = \begin{array}{c} \\ LP_1 \\ LP_2 \\ LP_3 \\ \vdots \end{array} \begin{array}{c} \begin{array}{ccccc} c_1 & c_2 & c_3 & \dots & m_N \end{array} \\ \begin{pmatrix} w_{1,1} & w_{1,2} & w_{1,3} & \dots & n_1 \\ w_{2,1} & w_{2,2} & w_{2,3} & \dots & n_2 \\ w_{3,1} & w_{3,2} & w_{3,3} & \dots & n_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \end{array} \tag{4.2}$$

A global synchronic distance value $GSV = \max_{f_1,f_2 \subset T} \{\sigma(f_1, f2)\}$ where T consists of the set of all transitions. The global synchronic distance is used to define a normalized synchronization measure. The global synchronization measure is the maximum synchronic distance in a P/T network and $\sigma_n(I_1, I_2) \in [0, 1]$ is a normalized synchronization value shown in Equation 4.3 where $\{I_n\}$ is a set of all incoming transitions to a particular place. A probability of being within tolerance is defined in vector $\vec{p}$ shown in Matrix 4.4. Each LP_i along the side of Matrix 4.4 indicates a Logical Process (LP) and the $1 - P_{ot}$ along the top of Matrix 4.4 indicates p_i values that are the individual probabilities that the tolerance is not exceeded. The probability of out-of-tolerance rollback is discussed in more detail in Section 4.3.1. Let (LP_i, c_j) be the transition from LP_i across connection c_j. After each transition of M_N from (LP_i, c_j), the next value of n_i which is the element in the i^{th} row of the last column of M_N is $\sigma_n(I_1, I_2){p_i}^{n_i}$.

$$\sigma_n(I_1, I_2) = 1.0 - \frac{\sigma(I_1, I_2)}{GSV} \tag{4.3}$$

$$\vec{p} = \begin{array}{c} \\ LP_1 \\ LP_2 \\ LP_3 \\ \vdots \end{array} \begin{array}{c} 1 - P_{ot} \\ \left(\begin{array}{c} p_1 \\ p_2 \\ p_3 \\ \vdots \end{array}\right) \end{array} \tag{4.4}$$

As $p_i^{n_i}$ approaches zero, the likelihood of an out-of-tolerance induced rollback increases. As $\sigma_n(I_1, I_2)p_i^{n_i}$ becomes very small, the likelihood of a rollback increases either due to violation of causality or an out-of-tolerance state value. The $\sigma_n(I_1, I_2)$ value is treated as a probability because it has the axiomatic properties of a probability. The axiomatic properties are that $\sigma_n(I_1, I_2)$ assigns a number greater than or equal to zero to each synchronic value, $\sigma_n(I_1, I_2)$ has the value of one when messages are always in order, and $\sigma_n(A) + \sigma_n(B) = \sigma_n(A \cup B)$ where A and B are mutually exclusive sets of transitions.

A brief example is shown in Figure 4.5. The initial state shown in Figure 4.5 is represented in Matrix 4.5. The Global Synchronic Distance of this network is four. The tolerance vector for this example is shown in Vector 4.6. Consider transition a shown in Figure 4.5; it is enabled since tokens are available in all of its inputs. The element in the $\vec{p}$ column vector shown in Vector 4.6 is taken to the power of the corresponding elements of the column vector $\vec{a}$ in Matrix 4.5 that are greater than zero ($p_i^{n_i}$). This is the probability that all messages passing through transition a arrive within tolerance. All columns of rows of $\vec{a}$ that are greater than zero that have greater than zero values form the input set ($\{I_n\}$) for $\sigma_n(I_1, I_2)$. Since transition a has only one input, $\sigma_n(\{a\})$ is one. When transition a fires, column vector $\vec{a}$ is added to column vector $\vec{m_0}$ to generate a new vector $\vec{m_1}$. Matrix 4.7 results after transition a fires. Continuing in this manner, Matrix 4.8 shows the result after transition b fires. Since $\sigma_n(\{b\})$ is one, row LP_4 of $\vec{m_2}$ is 0.3.

$$M_0 = \begin{array}{c} \\ LP_1 \\ LP_2 \\ LP_3 \\ LP_4 \\ LP_5 \\ LP_6 \\ LP_7 \\ LP_8 \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & m_0 \end{array} \\ \left(\begin{array}{ccccccc} 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{array}\right) \end{array} \tag{4.5}$$

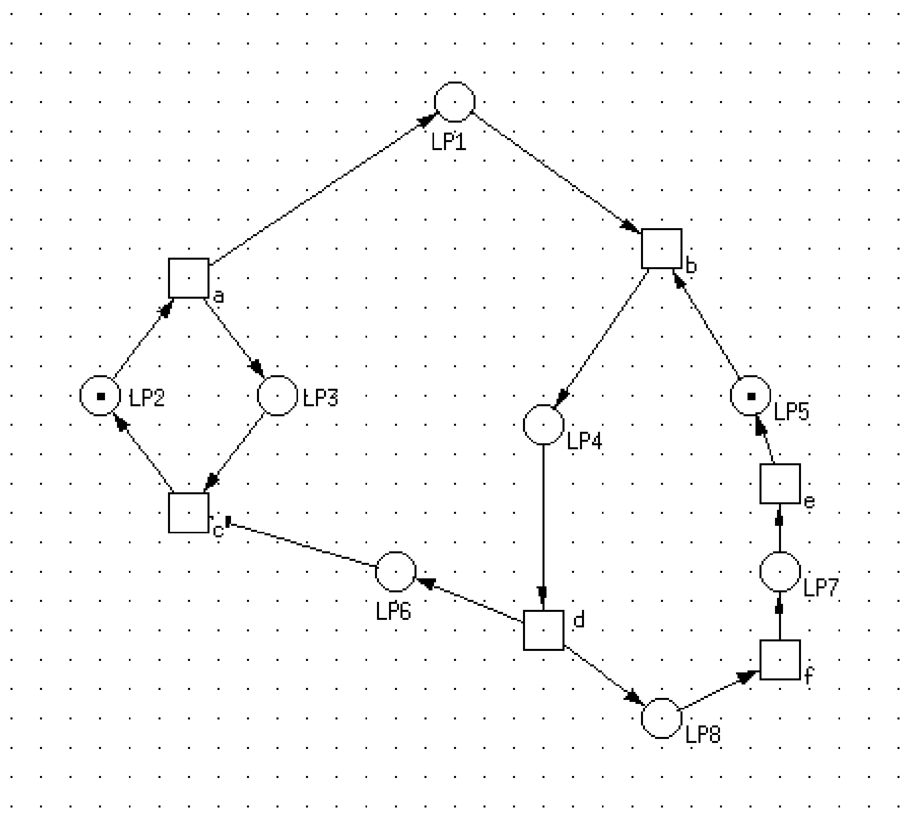


Figure 4.5: Example of P_{oo} Analysis.

$$\vec{p} = \begin{array}{c} \\ LP_1 \\ LP_2 \\ LP_3 \\ LP_4 \\ LP_5 \\ LP_6 \\ LP_7 \\ LP_8 \end{array} \begin{array}{c} 1 - P_{ot} \\ \left(\begin{array}{c} 0.7 \\ 0.2 \\ 0.3 \\ 0.4 \\ 0.6 \\ 0.4 \\ 0.2 \\ 0.1 \end{array}\right) \end{array} \tag{4.6}$$

$$M_1 = \begin{array}{c} \\ LP_1 \\ LP_2 \\ LP_3 \\ LP_4 \\ LP_5 \\ LP_6 \\ LP_7 \\ LP_8 \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & m_1 \end{array} \\ \left(\begin{array}{ccccccc} 1 & -1 & 0 & 0 & 1 & 0 & 0.7 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0.3 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{array}\right) \end{array} \tag{4.7}$$

$$M_2 = \begin{array}{c} \\ LP_1 \\ LP_2 \\ LP_3 \\ LP_4 \\ LP_5 \\ LP_6 \\ LP_7 \\ LP_8 \end{array} \begin{array}{c} \begin{array}{ccccccc} a & b & c & d & e & f & m_2 \end{array} \\ \begin{pmatrix} 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0.3 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0.3 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \end{pmatrix} \end{array} \tag{4.8}$$

The analysis presented in this section reduces the time and topological complexities characteristic of more explicit time analysis methods to simpler and more insightful matrix manipulations. The method presented in this section is used in the following section to determine the probability of rollback due to out-of-order messages, $P_{oo} = 1 - \sigma_n(I_1, I_2)$.

Also, the worst case proportion of out-of-order messages (X) can be calculated as follows. The synchronic distance ($\sigma(I_1, I_2)$) is a measure of the maximum difference in the rate of firing among transitions. The maximum possible rate that $\sigma(I_1, I_2)$ can occur is the rate of the slowest firing transition in sets I_1, I_2. Thus, $E[X] \leqslant \min_{\{Transition \in I_1, I_2\}}\{rate(Transition)\}$ where $rate(I)$ is the rate at that transition I fires.

4.2.1 P/T Analysis of VNC

This section applies the synchronic measure analysis to the Rapidly Deployable Radio Network (RDRN) Network Control Protocol (NCP). Figure 4.1 and Figure 4.6 show the Condition Event Network Nets for an Edge Switch (ES) and Remote Node (RN). Figure 4.7 shows a simple example of the Petri-Net analysis applied directly to the Network Control Protocol. The net shown in Figure 4.7 is a combination of Figure 4.1 and Figure 4.6 with a *handoff* place added to connect the two nets. The equivalent net shown in Figure 4.8 was analyzed using the PROD tool described in [115]. Note that complementary places exist in this net for two reasons. First, the tool that executes the reachability analysis assumes infinite capacity places. In order to implement single token capacity, the complement of each place was added. In addition the complement place provides the logical equivalent of a negative condition that is useful for controlling net behavior. Additional handoff places (H) can be added to hold H tokens that represents an expected ratio of H **USER_POS** packets for each **HANDOFF** packet. Using Figure 4.7, a relative measure for the out-of-orderedness of messages arriving at each place can be determined. For example, messages arrive at the **RnUpdate** place from transitions j and c. The synchronic distance between j and c is one, and the normalized synchronic measure is $\sigma_n(\{j\},\{c\}) = 1 - \frac{1}{H}$. Therefore these messages will arrive in order with a high likelihood if H is large. However, messages arrive into the **Active** place of the Edge Switch from three different transitions, f, h, and i. Messages only arrive from transition f due to a re-initialization, re-initialization will only occur with multiple Edge Switchs and will not be considered in Figure 4.7. That leaves transitions h and i that have a normalized synchronic value of $\sigma_n(\{h\},\{i\}) = 1 - \frac{(H-1)}{(H+1)}$. Thus, there is a high probability that a **HANDOFF** message may arrive out-of-order relative to a return to the **Active** place from an **RnUpdate**, because, except for the initial handoff, the ratio of

USER_POS to **HANDOFF** messages will usually be high. Finally transitions a and c lead to the **Active** place for the Remote Node with a normalized synchronic value of $\sigma_n(\{a\},\{c\}) = 1 - \frac{(H-1)}{(H+1)}$. This example has shown how synchronic distance is used to determine the likelihood of virtual messages arriving out-of-order at an Logical Process and determines the rate of occurrence of out-of-order messages (X) that is simply the rate at that **HANDOFF** messages arrive at **RnUpdate**. Therefore, $E[X]$ is less than or equal to the rate at which **HANDOFF** messages arrive at **RnUpdate**.

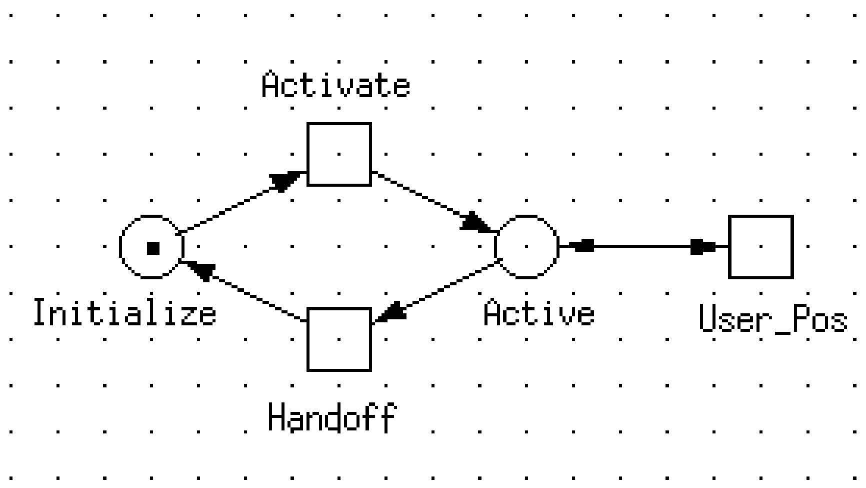


Figure 4.6: RN NCP FSM as a C/E Net.

4.3 Expected Speedup: η

This section analyzes the primary benefit of Virtual Network Configuration which is speedup. There are many factors which influence speedup including out-of-order message probability, out-of-tolerance state value probability, rate of virtual messages entering the system, task execution time, task partitioning into Logical Processs, rollback overhead, prediction accuracy as a function of distance into the future which predictions are attempted, and the effect of parallelism and optimistic synchronization. All

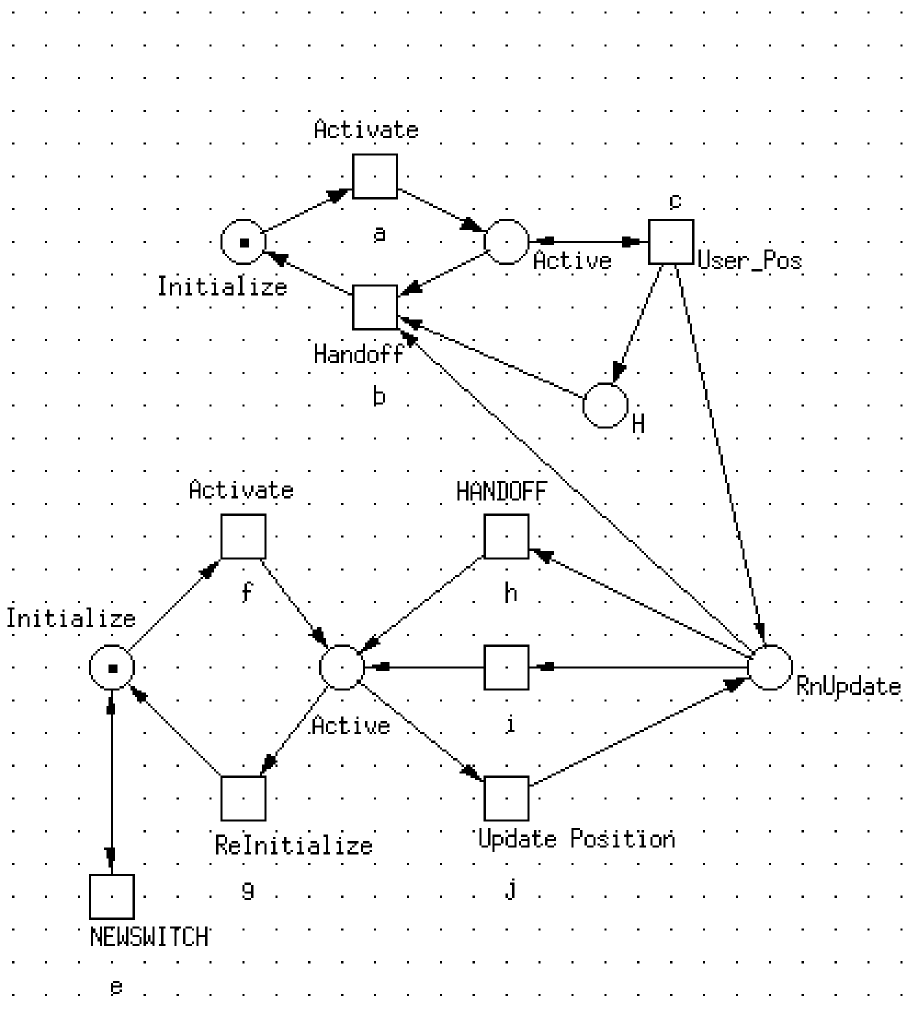


Figure 4.7: Example of RDRN Causal Analysis.

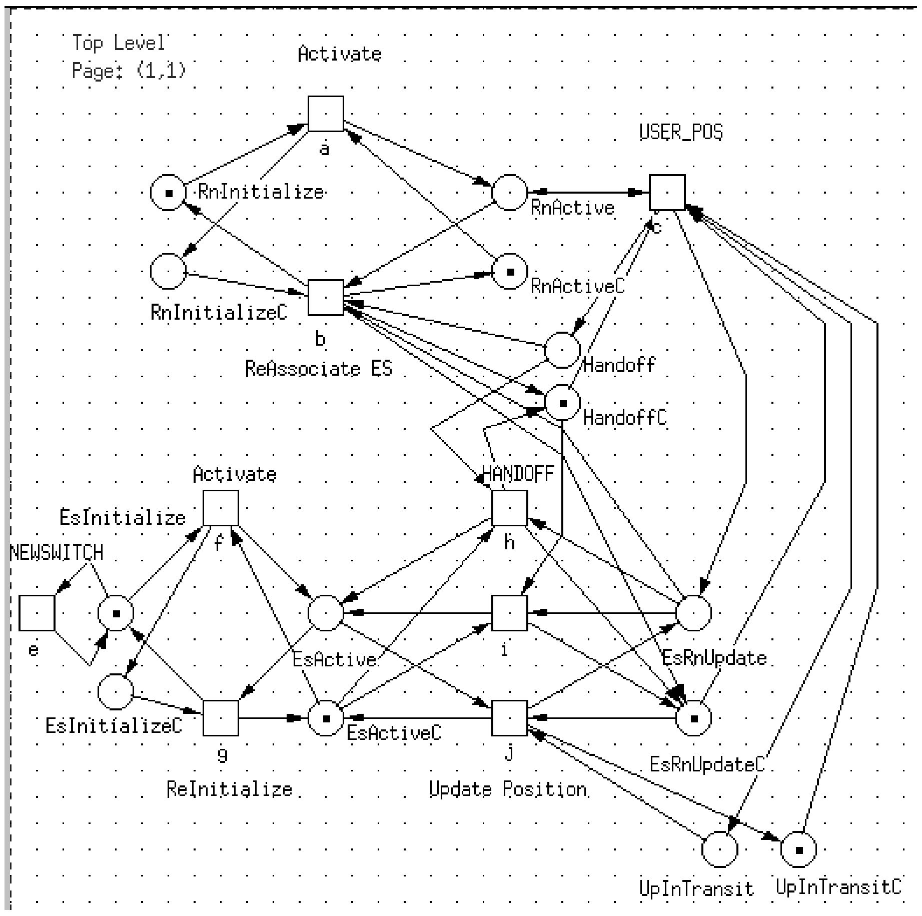


Figure 4.8: RDRN Causal Analysis with PROD.

of these factors are considered in this section beginning with a direct analysis using the definitions from optimistic simulation.

The definition of Global Virtual Time (GVT) can be applied to determine the relationship among expected task execution time (τ_{task}), the real time at which the state was cached (t_{SQ}), and real time (t). Consider the value (V_v) which is cached at real time t_{SQ} in the State Queue (SQ) resulting from a particular predicted event. For example, refer to Figures 3.8 through 3.10 and notice that state queue values may be repeatedly added and discarded as Virtual Network Configuration operation proceeds in the presence of rollback. As rollbacks occur, values for a particular predicted event may change, converging to the real value (V_r). For correct operation of Virtual Network Configuration, V_v should approach V_r as t approaches $GVT(t)$ where $GVT(t)$ is the GVT of the Virtual Network Configuration system at time t. Explicitly, this is $\forall \epsilon > 0 \; \exists \delta > 0$ s.t. $\mid f(t) - f(GVT(t)) \mid < \epsilon \Rightarrow 0 < \mid GVT(t) - t \mid < \delta$ where $f(t) = V_r$ and $f(GVT(t)) = V_v$. $f(t)$ is the prediction function of a driving process. The purpose and function of the driving process has been explained in Section 3.1. Because Virtual Network Configuration will always use the correct value when the predicted time (τ) equals the current real time (t) and it is assumed that the predictions will become more accurate as the predicted time of the event approaches the current time, the reasonable assumption is made that $\lim_{\tau \to t} f(\tau) = V_v$. In order for the Virtual Network Configuration system to always look ahead, $\forall \, t \; GVT(t) \geqslant t$. This means that $\forall \, n \in \{LPs\}$ and $\forall \, t \; LVT_{lp_n}(t) \geqslant t$ and $\min_{m \in \{M\}}\{m\} \geqslant t$ where m is the receive time of a message, M is the set of messages in the entire system and LVT_{lp_n} is the Local Virtual Time of the n^{th} Logical Process. In other words, the Local Virtual Time (LVT) of each Logical Process (LP) must be greater than or equal to real time and the smallest message Receive Time not yet processed must also be greater than or equal to real time. The smallest message Receive Time could cause a rollback to that time. This implies that

$\forall\, n, t\ LVT_{dp_n}(t) \geqslant t$. In other words, this implies that the Local Virtual Time (LVT) of each driving process must be greater than or equal to real time. An out-of-order rollback occurs when $m < LVT(t)$. The largest saved state time such that $t_{SQ} < m$ is used to restore the state of the Logical Process, where t_{SQ} is the real time the state was saved. Then the expected task execution time (τ_{task}) can take no longer than $t_{SQ} - t$ to complete in order for GVT to remain ahead of real time. Thus, a constraint between expected task execution time (τ_{task}), state save time (t_{SQ}), and real time (t) has been defined. The next section considers the effect of out-of-tolerance state values on the rollback probability and the concept of stability in Virtual Network Configuration.

4.3.1 Stability

Stability in Virtual Network Configuration is related to the ability of the system to reduce the number of rollbacks. An unstable system is one in which there exists enough rollbacks to cause the system to take longer than real-time to reach the end of the Sliding Lookahead Window (SLW). Rollback is caused by the arrival of a message which should have been executed in the past and by out-of-tolerance states. In either case, messages which had been generated prior to the rollback are false messages. Rollback is contained by sending anti-messages to cancel the effects of false messages. The more quickly the anti-messages can overtake the effect of false messages, the more efficiently rollback is contained.

Consider a specific scenario in the Rapidly Deployable Radio Network [17] of a single configuration processor on the mobile host and a single configuration processor on the base station. All processing must occur in strict sequential order and the only parallelism occurs between the processing on the mobile host and base station. The simplified feasibility analysis presented here assumes exponential processing times for each task. The system is driven by the location predictions as shown in Figure 4.9. Time lines which indicate virtual position updates beyond current time (t) are drawn in the figure below the corresponding processes. Λ is the length of the Sliding Lookahead Window ($SLW = (t, t + \Lambda]$) which extends from the end of current time ($t + \epsilon$) to $t + \Lambda$ and contains K position updates. There are N processes which process the K inputs. Examples of these processes are mobile host position prediction, beamforming, Private Network-Network Interface, IP Routing, Asynchronous Transfer Mode Address Resolution Protocol caching and Mobile IP configuration as shown in Figure 4.9.

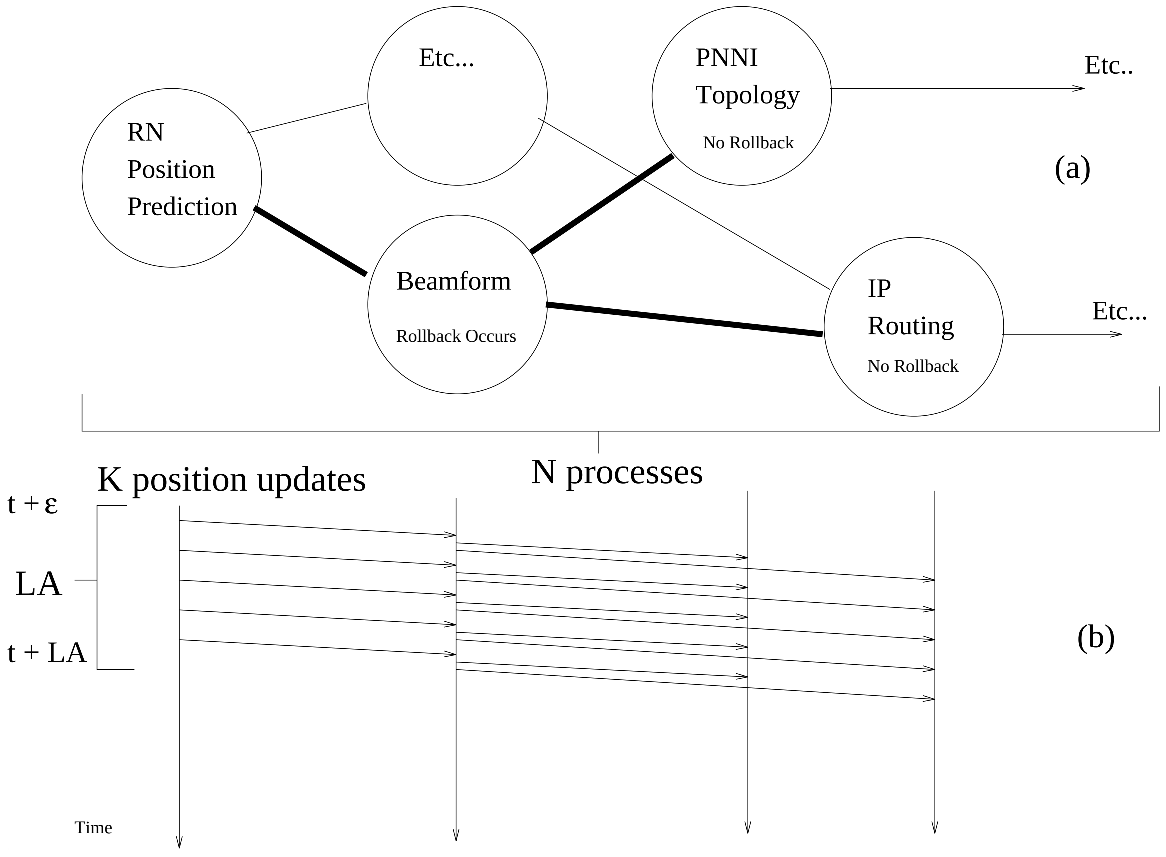


Figure 4.9: Relation Between Processes and Virtual Messages.

A cause of rollbacks in Virtual Network Configuration is real messages which are out of tolerance. Those processes which require a higher degree of tolerance are most likely to rollback. A worst case probability of out-of-tolerance rollback for a single process, shown in Equation 4.9, is based on Chebycheff's Inequality [85] from basic probability. The variance of the data is σ^2 and Θ is the acceptable tolerance for a configuration process. Therefore, the performance gains of Virtual Network Configuration are reduced as a function of P_{ot}. At the cost of increasing the accuracy of the driving process(es), that is, decreasing σ^2 in Proposition 1, P_{ot} becomes small thus increasing the performance gain of Virtual Network Configuration.

Proposition 1 *The probability of rollback of an Logical Process is*

$$P_{ot} \leqslant \frac{\sigma^2}{\Theta^2} \tag{4.9}$$

where P_{ot} is the probability of out-of-tolerance rollback for an Logical Process, σ^2 is the variance in the amount of error, and Θ is the tolerance allowed for error.

If the Kth position update predicts a handoff, then the total expected time for the prediction computation is $KN\frac{1}{\mu}$, where N is the number of processes and $\frac{1}{\mu}$ is the expected exponential processing time per process and there exists strict synchronization between processes.

The expected time between rollbacks for the Virtual Network Configuration system is critical for determining its feasibility. The probability of rollback for all processes is the probability of out-of-order message occurrence and the probability of out-of-tolerance state values ($P_{rb} = P_{oo} + P_{ot}$). The received message rate per Logical Process is R_m and there are N Logical Processs. The expected inter-rollback time for the system is shown in Equation 4.10.

Proposition 2 *The expected inter-rollback time is*

$$T_{rb} = \frac{1}{\lambda_{rb}} = \frac{1}{R_m N P_{rb}} \tag{4.10}$$

where T_{rb} is the expected inter-rollback time, λ_{rb} is the expected rollback rate, R_m is the received message rate per Logical Process, there are N Logical Processs, and P_{rb} is the probability of rollback per process.

4.3.2 Single Processor Logical Processes

The contribution of this section is the analysis of single processor logical process. Multiple Logical Processes on a single processor loose any gain in concurrency since they are being served by a single processor, however, the Logical Processes maintain the Virtual Network Configuration lookahead if partitioned properly. The Network Control Protocol's link management daemon code runs as a multiple process system on single processors. In the current architecture[1] each remote node link management daemon consists of three processes: a position prediction process, a control and Simple Network Management Protocol (SNMP) agent process, and a beamform and stack activation process. Each edge switch link management daemon consists of two processes: a control and Simple Network Management Protocol agent process, and beamform and stack activation process. Thus there are logical processes communicating on single processors and across multiple processors. Multiple Logical Processes executing on a single processor will not have speedup due to parallelism because a single processor must serve all the Logical Processes. Because these Logical Processes reside on a single processor, they are not operating in parallel as Logical Processes are assumed to do

[1]The revision control system identifier for Network Control Protocol code at the time this document was written was *esctrld.c,v 1.7 1997/02/01 20:31:25* and *rnctrld.c,v 1.8 1997/02/01 20:31:25*.

in an optimistic system, thus a new term needs to be applied to a task partitioned into Logical Processes on a single processor. Each partition of tasks into Logical Processes on a single processor is called a Single Processor Logical Process (SLP). In the upper portion of Figure 4.10, a task has been partitioned into two logical processes. The same task exists in the lower portion of Figure 4.10 as a single Logical Process. If task B must rollback because of an out-of-tolerance result, the entire single Logical Process must rollback, while only the Logical Process for task B must rollback in the multiple Logical Process case. Thus partitioning a task into multiple Logical Processes can save time compared to a single Logical Process task. Thus, **without considering parallelism**, lookahead can be achieved by allowing the sequential system to work ahead while individual tasks within the system are allowed to rollback. Only tasks which deviate beyond a given preconfigured tolerance are rolled back. Thus entire pre-computed and cached results are not lost due to inaccuracy, only parts of pre-computed results must be re-computed. This section discusses single processor logical processes, the next section reviews multiple processor logical processes, and a third section discusses the combination of a single/multiple processor system.

Consider the optimal method of partitioning a single processor system into Single Processor Logical Process (SLP)es in order to obtain speedup over a single Logical Process process. Assume n tasks, $task_1, ..., task_n$, with expected execution times of $\tau_1, ..., \tau_n$, and that $task_n$ depends on messages from $task_{n-1}$ with a tolerance of Θ_n. This is the largest error allowed in the input message such that the output is correct. Using the results from Proposition 1, it is possible to determine a partitioning of tasks into logical processes such that speedup is achieved over operation of the same tasks encapsulated in a single Logical Process. Figure 4.11 shows possible groupings of the same set of six tasks into logical processes. It is hypothesized that the tasks most likely to rollback and which take the greatest amount of time to execute should be

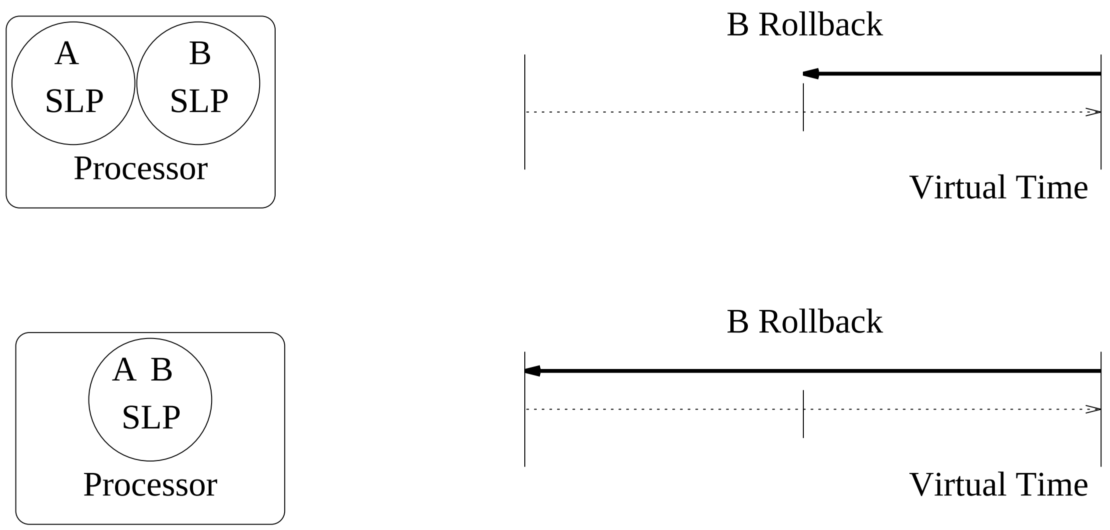


Figure 4.10: Single and Multiple Processor Logical Process System.

grouped together within Single Processor Logical Processes to minimize the rollback time. There are 2^{n-1} possible groupings of tasks into Single Processor Logical Processes where n is the number of tasks and message dependency among the tasks is maintained. Those tasks least likely to rollback and those which execute quickly should be grouped within a single Single Processor Logical Process to reduce the overhead of rollback. For example, if all the tasks in Figure 4.11 have an equal probability of rollback and $\tau_2 \gg \max\{\tau_1, \tau_3, ...\}$ then the tasks should be partitioned such that $task_2$ is in a separate SLP: $(task_1 \mid task_2 \mid task_3 ... task_n)$ where "|" indicates the grouping of tasks into sequential logical processes.

For example, the expected Logical Process execution time for five tasks with equal probabilities of rollback of 0.1 are shown in Figure 4.12. It is assumed that these tasks communicate in order starting from Task 1 to Task 5 in order to generate a result. In Figure 4.12, the x-axis indicates the boundary between task partitions as the probability

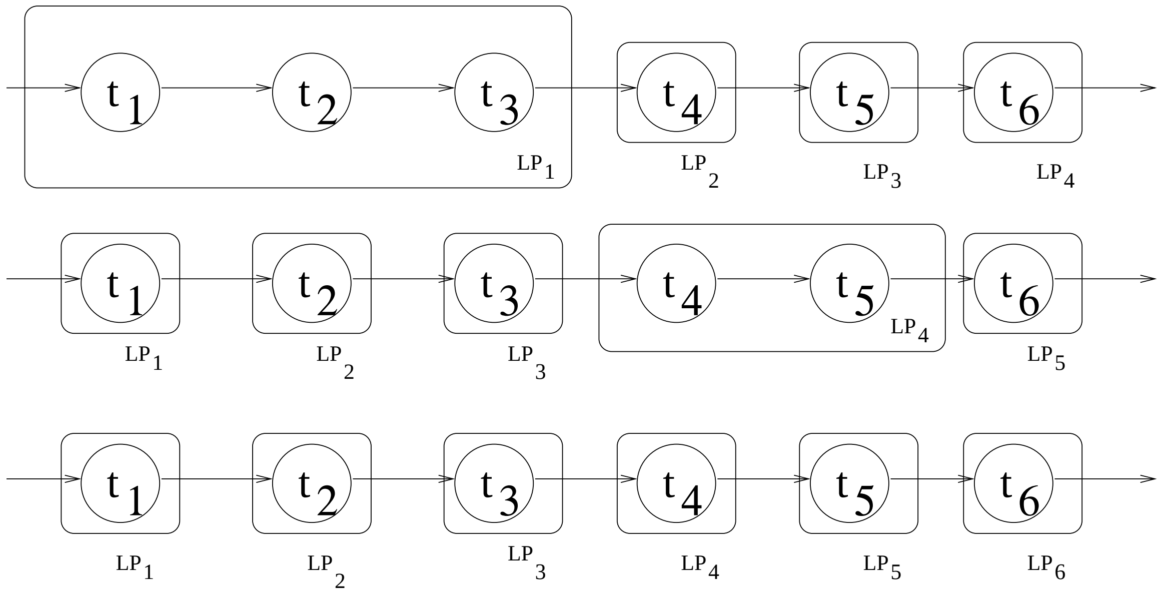


Figure 4.11: Possible Partitioning of Tasks into Logical Processes on a Single Processor.

of rollback of task 5 is varied. With an x-value of 3, the solid surface shows the expected execution time for the first three tasks combined within a single Logical Process and the the remainder of the tasks encapsulated in separate Logical Processes. The dashed surface shows the first three tasks encapsulated in separate Logical Processes and the remainder of the tasks encapsulated within an Logical Process. The graph in Figure 4.12 indicates a minimum for both curves when the high probability rollback tasks are encapsulated in separate Logical Processes from the low probability of rollback tasks. As the probability of rollback increases, the expected execution time for all five processes is minimized when Task 5 is encapsulated in a separate Logical Process.

Rapidly Deployable Radio Network Network Control Protocol Task Partition Analysis

Consider an example from the Rapidly Deployable Radio Network (RDRN) Network Control Protocol (NCP) operation. Assume the beamform table calculation time is

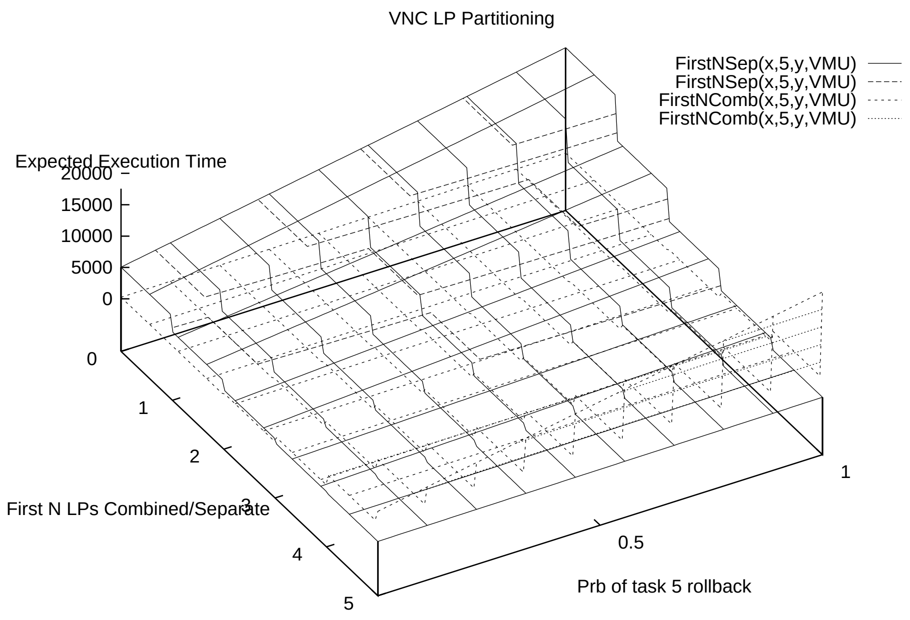


Figure 4.12: Optimal Single Processor Logical Process Partitioning.

exponentially distributed with mean $\frac{1}{\mu_{bT}}$. Assume the stack activation operation is also exponentially distributed with mean $\frac{1}{\mu_h}$. Beamforming has a rollback probability of P_{bT} and takes time τ_{bT} to rollback. Also, stack activation time has a rollback probability of P_h and takes time τ_h to rollback. If both operations are encapsulated by a single logical process, then the expected time of operation is shown in Equation 4.11. If each operation is encapsulated in a separate LP, then the expected time is shown in Equation 4.12. Equations 4.11 and 4.12 are formed by the sum of the expected time to execute the task which is the first term and the rollback time which is the second term. The probability of rollback in the combined Logical Process is the probability that either task will rollback. Therefore, the expected execution time of the tasks encapsulated in separate Logical Processes is smaller since $\tau_{separate} < \tau_{combined}$.

The grouping of tasks into Single Processor Logical Processes can be done dynamically, that is, while the system is in operation. This dynamic adjustment is currently

$$\tau_{combined} = (\frac{1}{\mu_{bT}} + \frac{1}{\mu_h}) + (\frac{1}{\mu_{bT}} + \frac{1}{\mu_h})(P_{bT} + P_h)(\tau_{bT} + \tau_h) \tag{4.11}$$

$$\tau_{separate} = (\frac{1}{\mu_{bT}} + \frac{1}{\mu_{bT}} P_{bT}\tau_{bT}) + (\frac{1}{\mu\, h} + \frac{1}{\mu\, h} P_h \tau_h) \tag{4.12}$$

outside the scope of this research but related to optimistic simulation load balancing [36, 37] and the recently developed topic of optimistic simulation dynamic partitioning [12, 60].

4.3.3 Single Processor Logical Process Prediction Rate

The Local Virtual Time (LVT) is a particular Logical Process's notion of the current time. In optimistic simulation the Local Virtual Time of individual processes may be different from one another and generally precede at a much faster rate than real time. Thus, the rate at which a Single Processor Logical Process (SLP) system can predict events (prediction rate) is the rate of change of the Single Processor Logical Process's Local Virtual Time with respect to real time. Assume a driving process whose virtual message generation rate is λ_{vm}. The Local Virtual Time is increased by the expected amount Δ_{vm} every $\frac{1}{\lambda_{vm}}$ time units. The expected time spent executing the task is τ_{task}. The random variables X and Y are the proportion of messages which are out-of-order and out-of-tolerance respectively. The expected real time to handle a rollback is τ_{rb}. Then the Single Processor Logical Process's Local Virtual Time advances at the expected rate shown in Proposition 3.

Proposition 3 (Single Processor Logical Process Speed) *The average prediction rate of a single logical processor system is*

$$S_{cache} = \frac{LVT}{t} = \lambda_{vm}(\Delta_{vm} - \tau_{task} - (\tau_{task} + \tau_{rb})E[X] - (\Delta_{vm} - \frac{1}{\lambda_{vm}})E[Y]) \tag{4.13}$$

where the virtual message generation rate is λ_{vm}, the expected lookahead per message is Δ_{vm}, the proportion of out-of-order messages is X, the proportion of out-of-tolerance messages is Y, τ_{task} is the expected task execution time in real time, τ_{rb} is the expected rollback overhead time in real time, LVT is the Local Virtual Time (LVT), and t is real time.

In Proposition 3 the expected lookahead per message (Δ_{vm}) is reduced by the real time taken to process the message (τ_{task}). The expected lookahead is also reduced by the time to re-execute the task (τ_{task}) and the rollback time (τ_{rb}) times the the proportion of occurrences of an out-of-order message ($E[X]$) which results in the term $(\tau_{task} + \tau_{rb})E[X]$. Finally, the derivation of the ($\Delta_{vm} - \frac{1}{\lambda_{vm}})E[Y]$ term is shown in Figure 4.13. In Figure 4.13, a real message arrives at time t which is labeled in Figure 4.13. Note that real time t and Local Virtual Time (LVT) are both shown on the same time axis in Figure 4.13. The current Local Virtual Time of the process is labeled at time LVT(t) in Figure 4.13. The dotted line in Figure 4.13 represents the time $\Delta_{vm} - \frac{1}{\lambda_{vm}}$ which is subtracted from the Local Virtual Time when an out-of-tolerance rollback occurs. The result of the subtraction of $\Delta_{vm} - \frac{1}{\lambda_{vm}}$ from the $LVT(t)$ results in the Local Virtual Time returning to real time as required by the algorithm. The virtual message inter-arrival time is $\frac{1}{\lambda_{vm}}$. Note that the ($\Delta_{vm} - \frac{1}{\lambda_{vm}})E[Y]$ term causes the speedup to approach one based on the frequency of out-of-tolerance rollback ($E[Y]$).

In the specific case of the Rapidly Deployable Radio Network (RDRN) Remote Node (RN) beamforming Single Processor Logical Process with no prediction error the expected prediction rate is 3.5 virtual seconds per second given $\lambda_{vm} = 0.1$ virtual messages per millisecond, $\Delta_{vm} = 45.0$ milliseconds, $\tau_{task} = 10.0$ milliseconds, $\tau_{rb} = 1.0$ milliseconds. The speedup quantified in Equation 4.13 is labeled S_{cache} because there is also a synchronization speedup, $S_{parallel}$ which is defined later.

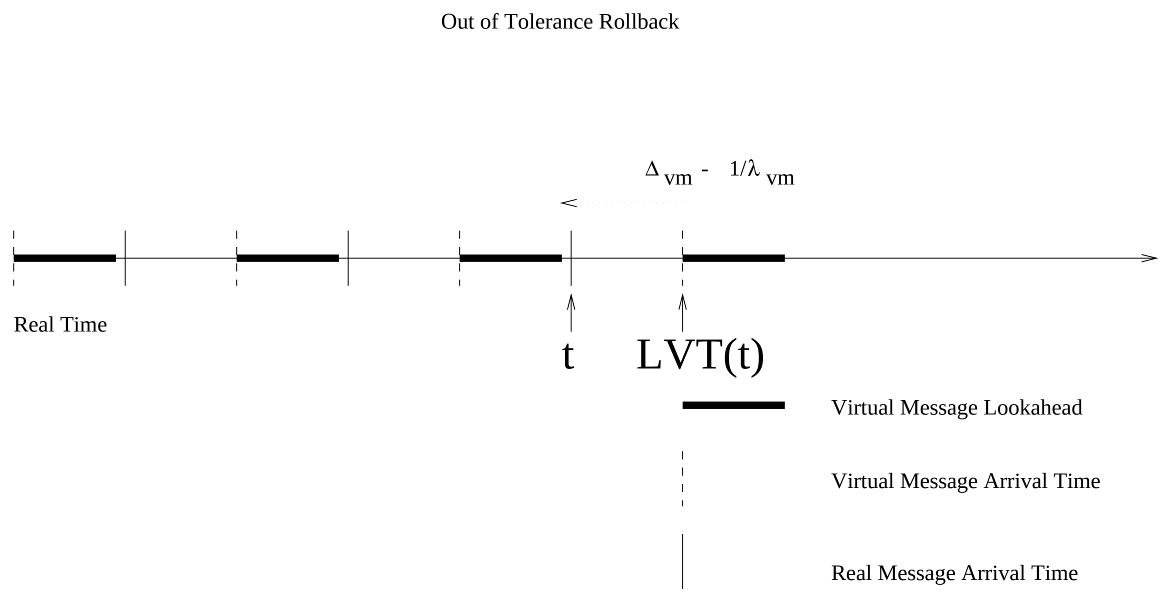

Out of Tolerance Rollback
Δ vm - 1/λ vm
Real Time
t
LVT(t)
Virtual Message Lookahead
Virtual Message Arrival Time
Real Message Arrival Time

Figure 4.13: Out-of-Tolerance Rollback.

4.3.4 Sensitivity

If the proportion of out-of-tolerance messages, Y, cannot be reduced to zero, the virtual message generation rates and expected virtual message lookahead times can be adjusted in order to improve speedup. Given the closed form expression for Virtual Network Configuration speedup in Proposition 3, it is important to determine the optimal values for each parameter, particularly λ_{vm} and Δ_{vm} and in addition, the sensitivity of each parameter. Sensitivity information will indicate which parameters most affect the speedup. The parameters which most affect the speedup are the ones that will yield the best results if optimized.

$$\begin{aligned} \nabla f(x^*) + \lambda^T \nabla h(x^*) + \mu^T \nabla g(x^*) &= 0 \\ \mu^T g(x^*) &= 0 \\ \mu &\geqslant 0 \end{aligned} \tag{4.14}$$

A technique which optimizes a constrained objective function and which also determines the sensitivity of each parameter within the constraints is the Kuhn-Tucker method [75, p. 314]. The reason for using this method rather than simply taking the derivative of Equation 4.13 is that the optimal value must reside within a set of constraints. Depending on the particular application of Virtual Network Configuration (VNC), the constraints may become more complex than those shown in this example. The constraints for this example are discussed in detail later. The sensitivity results appear as a by-product of the Kuhn-Tucker method. The first order necessary conditions for an extremum using the Kuhn-Tucker method are listed in Equation 4.14. The second order necessary conditions for an extremum are given in Equation 4.15 where L must be positive semi-definite over the active constraints and L, F, H, and G are Hessians. The second order sufficient conditions are the same as the first order necessary conditions and the Hessian matrix in Equation 4.15 is positive definite on the subspace $M = \{y : \nabla h(x)y = 0, \nabla g_j(x)y = 0 \text{ for all } j \in J\}$, where $J = \{j : g_j(x) = 0, \mu_j \geqslant 0\}$. The sensitivity is determined by the Lagrange multipliers, λ^T and μ^T. The Hessian of the objective function and of each of the inequality constraints is a zero matrix, thus, the eigenvalues L in Equation 4.15 are zero and the matrix is clearly positive definite satisfying both the necessary and sufficient conditions for an extremum.

The function f in Equation 4.14 is the Virtual Network Configuration speedup given in Equation 4.13. The matrix h does not exist, because there are no equality constraints,

$$L(x^*) = F(x^*) + \lambda^T\, H(x^*) + \mu^T\, G(x^*) \tag{4.15}$$

and the matrix g consists of the inequality constraints which are specified in Equation 4.17. The bounds chosen in Equation 4.17 are based on measurements from a specific application, Rapidly Deployable Radio Network configuration. Clearly the upper bound constraints on $E[X]$ and $E[Y]$ are the virtual message rate. The constraints for τ_{task} and τ_{rb} are based on measurements of the task execution time and the time to execute a rollback. The maximum value for λ_{vm} is determined by the rate at which the virtual message can be processed. Finally, the maximum value for Δ_{vm} is determined by the required caching period. If Δ_{vm} is too large, there may be no state in the State Queue (SQ) with which to compare an incoming real message.

From inspection of Equation 4.13 and the constraint shown in Equation 4.16, the constraints from 4.17 are $\Delta_{vm} = 45.0$, $\tau_{task} = 5.0$, $\tau_{rb} = 1.0$, $E[X] = 0.0$, $E[Y] = 0.0$ which results in the optimal solution shown in Equations 4.19. The Lagrange multipliers μ_1 through μ_6 show that $E[Y]$ $(-\,\mu_6 = -8.0)$, λ_{vm} $(-\,\mu_1 = -40.0)$, and $E[X]$ $(-\,\mu_5 = -1.2)$ have the greatest sensitivities. Therefore, reducing the out-of-tolerance rollback has the greatest effect on speedup. In the next section the effect of optimistic synchronization on speedup is derived.

$$\lambda_{vm} = \frac{1}{\tau_{task}+(\tau_{task}+\tau_{rb})E[X]+\left(\Delta_{vm}-\frac{1}{\lambda_{vm}}\right)E[Y]} \tag{4.16}$$

$$0.0 \leqslant \lambda_{vm} \leqslant \frac{1}{\tau_{task}+(\tau_{task}+\tau_{rb})E[X]+\left(\Delta_{vm}-\frac{1}{\lambda_{vm}}\right)E[Y]} \tag{4.17}$$

$$0.1 \leqslant \Delta_{vm} \leqslant 45.0 \tag{4.18}$$

$$5.0 \leqslant \tau_{task} \leqslant 10.0$$

$$1.0 \leqslant \tau_{rb} \leqslant 2.0$$

$$0.0 \leqslant E[X] \leqslant 1.0$$

$$0.0 \leqslant E[Y] \leqslant 1.0$$

$$\lambda_{vm} = 1.0, \mu_1 = 40.0 \tag{4.19}$$

$$\Delta_{vm} = 45.0, \mu_2 = 0.2$$

$$\tau_{task} = 0.0, \mu_3 = 0.2$$

$$\tau_{rb} = 0.0, \mu_4 = 0.0$$

$$E[X] = 0.0, \mu_5 = 1.2$$

$$E[Y] = 0.0, \mu_6 = 8.0$$

4.3.5 Sequential Execution Multiple Processors

A comparison of optimistic synchronization with sequential synchronization has not been found in the literature because little work has been done which combines optimistic synchronization with a real time system with the exception of hybrid systems such as the system described in [7]. The hybrid system described in [7] is used as a design technique in which distributed simulation Logical Process (LP)s are gradually replaced with real system components allowing the emulated system to be executed as the system is built. It does not focus on predicting events as in Virtual Network

Configuration. This section examines sequential execution of tasks which corresponds with non-Virtual Network Configuration operation as shown in Figure 4.14 in order to compare it with the Virtual Network Configuration algorithm in the next section. As a specific example related to predicting handoff, consider the case of K position updates with P processes required for each handoff and each process has an exponential processing time with average $\frac{1}{\mu}$. In the sequential case, the expected completion time should be K times the summation of P exponential distributions. The summation of P exponential distributions is a Gamma Distribution as shown in the sequential execution pdf in Equation 4.20. The average time to complete K tasks is shown in Equation 4.21.

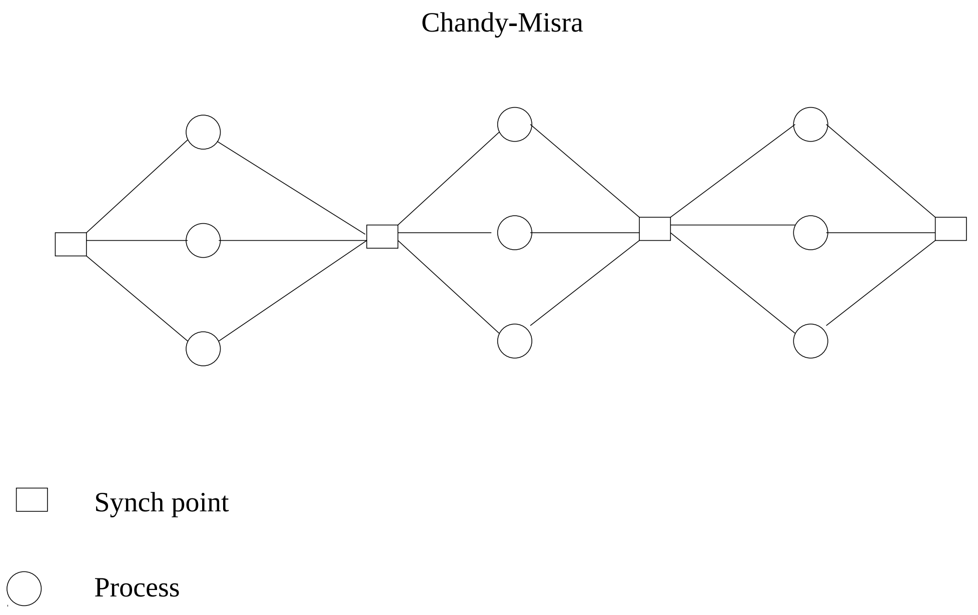


Figure 4.14: Sequential Model of Operation.

$$f_T(x \mid P, \mu) = \begin{cases} \frac{\mu^P}{\Gamma(P)} x^{P-1} \exp^{-\mu x} & x > 0 \\ 0 & x \leqslant 0 \end{cases} \tag{4.20}$$

$$T_{seq} = K \int_0^\infty x f_T(x \mid P, \mu)\, dx \tag{4.21}$$

4.3.6 Asynchronous Execution Multiple Processors

Assume that ordering of events is no longer a requirement. This represents the asynchronous Virtual Network Configuration case and is shown in Figure 4.15. Note that this is the analysis of speedup due to parallelism only, not the lookahead capability of Virtual Network Configuration. This analysis of speedup assumes messages arrive in correct order and thus there is no rollback. However, this also assumes that there are no optimization methods such as lazy cancellation. Following [32] the expected completion time is approximated by the maximum of P K-stage Erlangs where P is the number of processes which can execute in parallel at each stage of execution. A K-stage Erlang model represents the total service time as a series of exponential service times, where each service time is performed by a process residing on an independent processor in this case. There is no need to delay processing within the K-stage model because of inter-process dependencies, as there is for synchronous and sequential cases. For the specific case of Rapidly Deployable Radio Network (RDRN) Network Control Protocol (NCP) there would usually be two processes operating in parallel, the Edge Switch (ES) and Remote Node (RN). However, there can be three Logical Processs operating in parallel when operations between the Master Edge Switch, and a particular associated Edge Switch and Remote Node are required, or between an Edge Switch, an Remote Node, and a destination Edge Switch in a handoff. The most time consuming operations such as the Edge Switch to Edge Switch topology calculation, if partitioned in such a way as to minimize bandwidth overhead, can be executed in parallel on all

Edge Switchs. Equation 4.22 shows the pdf for a K-stage Erlang distribution.

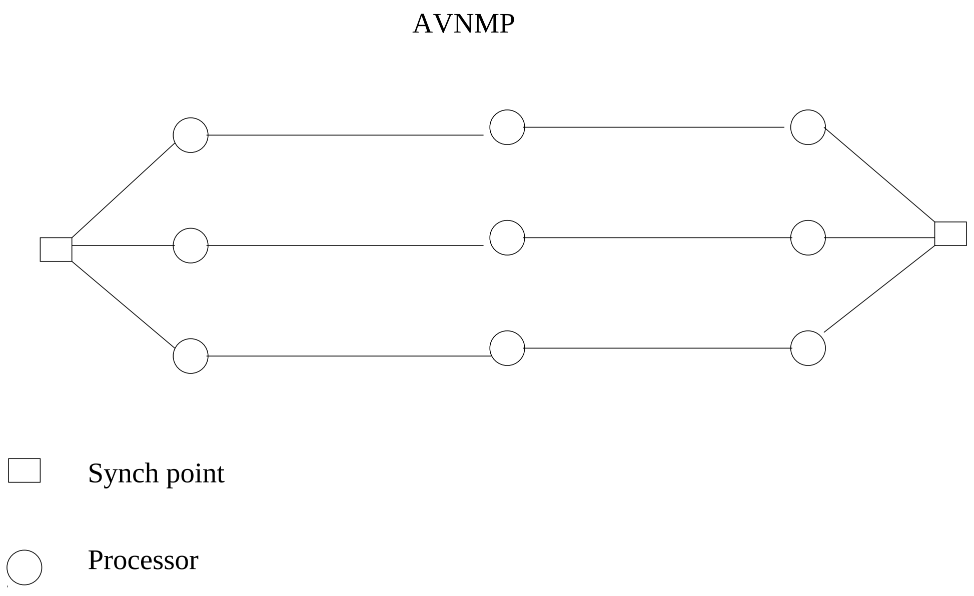


Figure 4.15: VNC Model of Parallelism.

$$f_T(x) = \frac{\mu\, e^{-\mu\, x} (\mu\, x)^{K-1}}{(K-1)!} \tag{4.22}$$

As pointed out in [32], the probability that a K-stage Erlang takes time less than or equal to t is one minus the probability that it takes time greater than t, which is simply one minus the probability that there are K arrivals in the interval $[0, t]$ from a Poisson process at rate μ. This result is shown in Equation 4.23.

The expected value is shown in Equation 4.24. This integral is hard to solve with a closed form solution and [32] instead tries to find an approximate equation. This

$$F_T(x) = 1 - e^{-\mu\, x} \sum_{i=0}^{K-1} \frac{(\mu\, x)^i}{i!} \tag{4.23}$$

$$T_{async} = \int_0^\infty [1 - F_T(x)]\, dx \tag{4.24}$$

study attempts to be exact by using Equation 4.24 and solving it numerically [59, p. 378]. In Equation 4.25 $S_{parallel}$ is the speedup of optimistic synchronization over strictly sequential synchronization and is graphed in Figure 4.16 as a function of the number of processors. The speedup gained by parallelism ($S_{parallel}$) augments the speedup due to lookahead (S_{cache}) as shown in Equation 4.26, where PR is the Virtual Network Configuration speedup and X and Y are random variables representing the proportion of out-of-order and out-of-tolerance messages respectively.

$$S_{parallel} = \frac{T_{seq}}{T_{async}} \tag{4.25}$$

The Rapidly Deployable Radio Network system consists of both Single Processor Logical Processs and LPs. Each Edge Switch (ES) and Remote Node (RN) executes network configuration processes (Single Processor Logical Process) and these processors communicate between nodes (Logical Process). Thus, there is the potential for speedup via parallel processing. The Virtual Network Configuration (VNC) algorithm implementation is able to take advantage of both Single Processor Logical Process lookahead without parallel processing and speedup due to parallel processing because Virtual Network Configuration has been implemented on many nodes throughout the network and each node has its own processor. Note that while Clustered Time Warp [4], which was developed concurrently but independently of Virtual Network Configuration, uses a similar concept to Single Processor Logical Process (SLP) and Logical Process (LP), it does not consider a real-time system as in Virtual Network Configu-

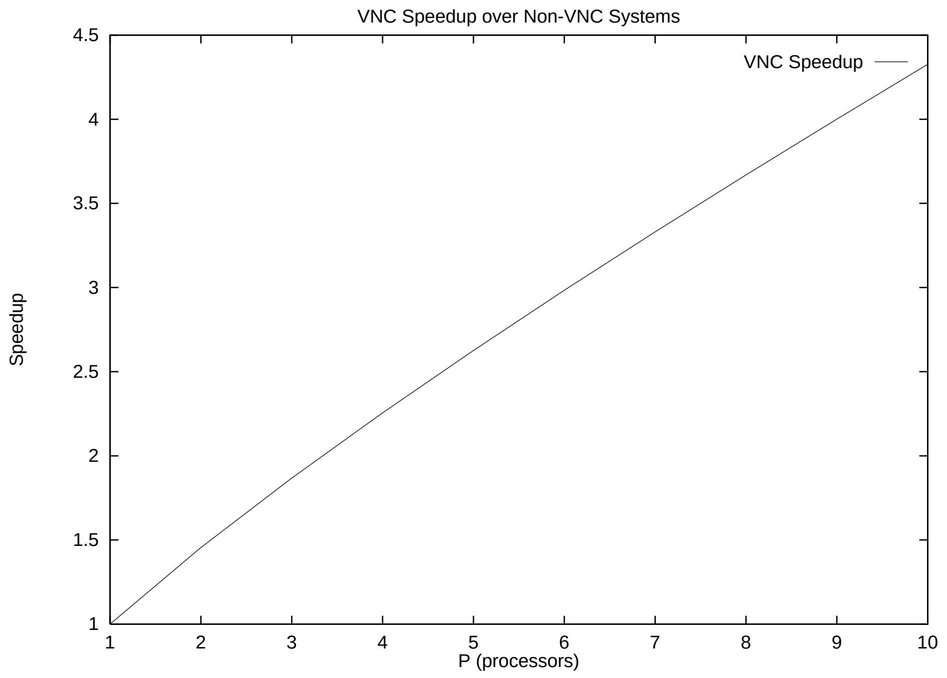


Figure 4.16: Speedup of VNC over Non-VNC Systems Due to Parallelism.

$$PR_{X,Y} = \lambda_{vm}\left(\Delta_{vm}S_{parallel} - \tau_{task} - (\tau_{task} + \tau_{rb})X - (\Delta_{vm}S_{parallel} - \frac{1}{\lambda_{vm}})Y\right) \tag{4.26}$$

ration. The next section considers the prediction rate given the maximum lookahead parameter Λ.

4.3.7 Multiple Processor Logical Processes

The goal of Virtual Network Configuration is to provide accurate predictions quickly enough so that the results are available before they are needed. Without taking advantage of parallelism, a less sophisticated algorithm than Virtual Network Configuration could run ahead of real-time and cache results for future use such as the Single Processor Logical Process system which assumes strict synchronization between processes whose prediction rate is defined in Proposition 3. With such a simpler mechanism, P_{oo} and $E[X]$ are always zero. However, simply predicting and caching results ahead of time does not fully utilize inherent parallelism in the system as long as messages between Logical Processs remain strictly synchronized. Strict synchronization means that processes must wait until all messages are insured to be processed in order. Any speedup to be gained through parallelism comes from the same mechanism as in optimistic parallel simulation; the assumption that messages arrive in order by Receive Time (TR), thus eliminating unnecessary synchronization delay. However, messages arrive out-of-order in Virtual Network Configuration for the following reasons. A general purpose system using the Virtual Network Configuration algorithm may have multiple driving processes, each predicting at different rates into the future. Another reason for out-of-order messages is that Logical Processs are not required to wait until processing completes before sending the next message. For example, a process which loads beam table information into memory need not wait for the load to complete before sending a virtual message. Also, processes may run faster for virtual computations by allowing a larger tolerance. Finally, for testing purposes, hardware or processes may be replaced with simulated code, thus generating results faster than the actual process would. Thus, although real and future time are working in parallel with strict synchronization, no advantage is being taken of parallel processing. This is demonstrated by the fact that, with strict synchronization of messages, the same speedup (S_{cache}) as defined in Proposition 3 occurs regardless of whether single processor or multiple processors are used. What differentiates Virtual Network Configuration is the fact that it takes advantage of inherent parallelism in the system than a sequential non-VNC pre-computation and caching method. Thus it is better able to meet the deadline imposed by predicting results before they are required. To see why this is true, consider what happens as the overhead terms in Proposition 3, $\tau_{task} - (\tau_{task} + \tau_{rb})E[X] - (\Delta_{vm} S_{parallel} - \frac{1}{\lambda_{vm}})E[Y]$, approach Δ_{vm}. The prediction rate becomes equal to real-time and can fall behind real-time as $\tau_{task} - (\tau_{task} + \tau_{rb})E[X] - (\Delta_{vm} S_{parallel} - \frac{1}{\lambda_{vm}})E[Y]$ becomes larger. Optimistic synchronization helps to alleviate the problem of the prediction rate falling behind real-time. Optimistic synchronization has another advantageous property, super-criticality. A super critical system is one which can compute results faster than the time taken by the critical path through the system. This can occur in Virtual Network Configuration using the lazy cancellation optimization as discussed in Section 3.1. Super-criticality occurs when task execution with false message values generate a correct result. Thus prematurely executed tasks do not rollback and a correct result is generated faster than the route through the critical path.

The Virtual Network Configuration algorithm has two forms of speedup which need to be clearly defined. There is the speedup in availability of results because they have been pre-computed and cached. There is also the speedup due to more efficient usage of parallelism. The gain in speedup due to parallelism in Virtual Network Configuration can be significant given the proper conditions. This can be achieved by neighboring Remote Node (RN) and Edge Switch (ES) nodes running on separate processors as they currently do, rather than physically adding multiple processors to each node. In order that there be no confusion as to which type of speedup is being analyzed, the speedup due to pre-computing and caching results will be defined as S_{cache} and the speedup due to parallelism will be defined as $S_{parallel}$. Speedup due to parallelism among multiple processors in Virtual Network Configuration will be gained from the same mechanism which provides speedup in parallel simulation, that is, it is assumed that all relevant messages are present and will be processed in order by receive time. The method of maintaining message order is optimistic in the form of rollback. The following sections look at $S_{parallel}$ due to a multiprocessor configuration system, such as the parallel operation of Remote Node and Edge Switch nodes.

4.3.8 Virtual Network Configuration Prediction Rate with a Fixed Lookahead

There are three possible cases to consider when determining the speedup of Virtual Network Configuration over non-lookahead sequential execution. In this section we will determine the speedup given each of these cases and their respective probabilities. These cases are illustrated in Figures 4.17 through 4.19. The time that an event is predicted to occur and the result cached is labeled $t_{virtual\ event}$, the time a real event occurs is labeled $t_{real\ event}$, and the time a result for the real event is calculated is labeled t_{no-vnc}. In Virtual Network Configuration, the virtual event and its result can be cached before the real event as shown in Figure 4.17, between the real event but before the real event result is calculated shown in Figure 4.18, or after the real event result is calculated as shown in Figure 4.19. In each case, all events are considered relative to the occurrence of the real event. It is assumed that the real event occurs at time t. A random variable called the lookahead (LA) is defined as $LVT - t$. The virtual event occurs at time $t - LA$. Assume that the task which must be executed once the real event occurs takes τ_{task} time. Then without Virtual Network Configuration the task is completed at time $t + \tau_{task}$.

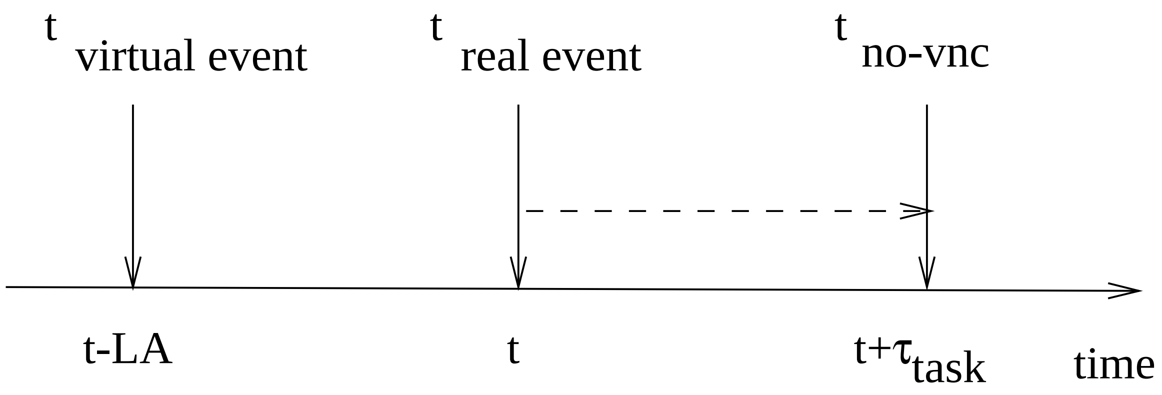


Figure 4.17: Virtual Network Configuration Cached before Real Event.

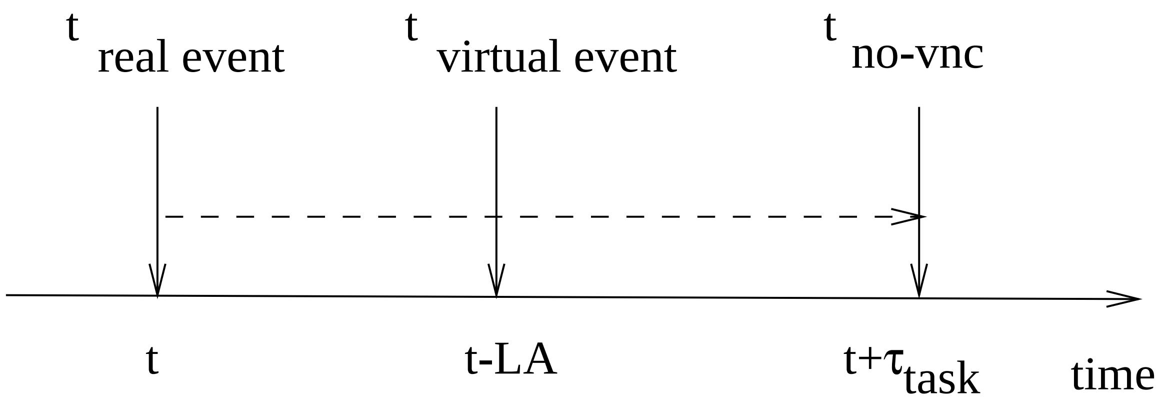


Figure 4.18: Virtual Network Configuration Cached later than Real Event.

The prediction rate has been defined in Equation 4.26 and includes the time to predict an event and cache the result in the State Queue (SQ). Recall that in Section 4.2 the expected value of X has been determined based on the inherent synchronization of the Logical Process topology. It was shown that X has an expected value which varies with the rate of hand-offs. Based on the topology of Network Control Protocol Logical Processs in Section 4.2, the virtual message path from the driving process to the

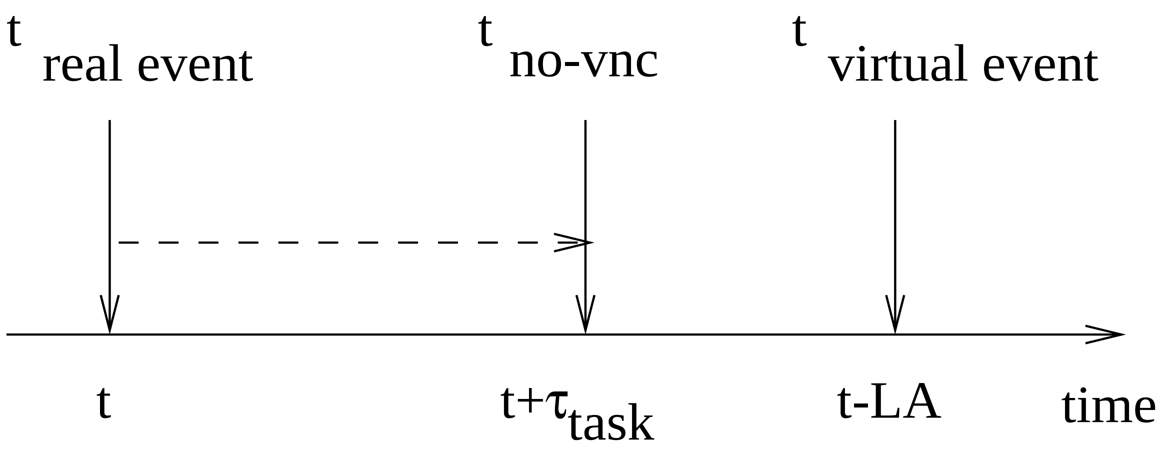


Figure 4.19: VNC Cached slower than Real Time.

beamforming Logical Process is only one level. Thus, there are no secondary rollbacks which could be generated. This means that the proportion of out-of-order rollback messages has a low variance. It is clear that the proportion of out-of-order messages is dependent on the Logical Process (LP) architecture and the partitioning of tasks into Logical Processs. Thus, it is difficult in an experimental implementation to vary X. It is easier to change the tolerance rather than change the Logical Process architecture to evaluate the performance of Virtual Network Configuration. For these reasons, the analysis proceeds with $PR_{X,Y|X=E[X]}$. Since the prediction rate is the rate of change of Local Virtual Time with respect to time, the value of the Local Virtual Time is shown in Equation 4.27 where C is an initial offset. This offset may occur because Virtual Network Configuration may begin running C time units before or after the real system. Replacing LVT in the definition of LA with the right side of Equation 4.27 yields the Equation for lookahead shown in Equation 4.28.

$$LVT_{X,Y|X=E[X]} = \tag{4.27}$$
$$\lambda_{vm}(\Delta_{vm}S_{parallel} - \tau_{task} - (\tau_{task} + \tau_{rb})E[X] -$$
$$(\Delta_{vm}S_{parallel} - \frac{1}{\lambda_{vm}})Y)t + C$$
$$LA_{X,Y|X=E[X]} = (LVT_{X,Y|X=E[X]} - 1)t + C \tag{4.28}$$

The probability of the event in which the Virtual Network Configuration result is cached before the real event is defined in Equation 4.29. The probability of the event for which the Virtual Network Configuration result is cached after the real event but before the result would have been calculated in the non-Virtual Network Configuration system is defined in Equation 4.30. Finally, the probability of the event for which the Virtual Network Configuration result is cached after the result would have been calculated in a non-Virtual Network Configuration system is defined in Equation 4.31.

$$P_{cache} = P[LA_{X,Y|X=E[X]} > \tau_{task}] \tag{4.29}$$
$$P_{late} = P[0 \leqslant LA_{X,Y|X=E[X]} \leqslant \tau_{task}] \tag{4.30}$$
$$P_{slow} = P[LA_{X,Y|X=E[X]} < 0] \tag{4.31}$$

The goal of this analysis is to determine the effect of the proportion of out-of-tolerance messages (Y) on the speedup of a Virtual Network Configuration system. Hence we assume that the proportion Y is a binomially distributed random variable with parameters n and p where n is the total number of messages and p is the probability of any single message being out of tolerance. It is helpful to simplify Equation 4.28 by using γ_1 and γ_2 as defined in Equations 4.33 and 4.34 in Equation 4.32.

$$LA_{X,Y|X=E[X]} = \gamma_1 - \gamma_2 Y \tag{4.32}$$

$$\gamma_1 = (\lambda_{vm} \Delta_{vm} S_{parallel} - \lambda_{vm} \tau_{task} - \lambda_{vm}(\tau_{rb} + \tau_{task})E[X]) - 1)t + C \tag{4.33}$$

$$\gamma_2 = \lambda_{vm}(\Delta_{vm} S_{parallel} - \frac{1}{\lambda_{vm}} + \tau_{rb})t \tag{4.34}$$

The early prediction probability as illustrated in Figure 4.17 is shown in Equation 4.35. The late prediction probability as illustrated in Figure 4.18 is shown in Equation 4.36. The probability for which Virtual Network Configuration falls behind real time as illustrated in Figure 4.19 is shown in Equation 4.37. The three cases for determining Virtual Network Configuration speedup are thus determined by the probability that Y is greater or less than two thresholds.

$$P_1(t) = P_{cache\ X,Y|X=E[X]} = P[Y < \tfrac{\gamma_1 - \tau_{task}}{\gamma_2}] \tag{4.35}$$

$$P_2(t) = P_{late\ X,Y|X=E[X]} = P[\tfrac{\gamma_1 - \tau_{task}}{\gamma_2} \leqslant Y \leqslant \tfrac{\gamma_1}{\gamma_2}] \tag{4.36}$$

$$P_3(t) = P_{slow\ X,Y|X=E[X]} = P[Y > \tfrac{\gamma_1}{\gamma_2}] \tag{4.37}$$

The three probabilities in Equations 4.35 through 4.37 depend on (Y) and real time because the analysis assumes that the lookahead increases indefinitely which shifts the thresholds in such a manner as to increase Virtual Network Configuration performance as real time increases. However, the Virtual Network Configuration algorithm holds processing of virtual messages once the end of the Sliding Lookahead Window (SLW) is reached. The hold time occurs when $LA = \Lambda$ where Λ is the length of the Sliding

Lookahead Window. Once Λ is reached, processing of virtual messages is discontinued until real-time reaches Local Virtual Time. The lookahead versus real time including the effect of the Sliding Lookahead Window is shown in Figure 4.20. The dashed arrow represents the lookahead which increases at rate PR. The solid line returning to zero is lookahead as the Logical Process delays. Because the curve in Figure 4.20 from 0 to t_L repeats indefinitely, only the area from 0 to t_L need be considered. For each $P_i(t)$ $i = 1, 2, 3$, the time average over the lookahead time (t_L) is shown by the integral in Equation 4.38.

$$P_{X,Y|X=E[X]} = \frac{1}{t_L} \int_0^{t_L} P_i(t)\,dt \tag{4.38}$$

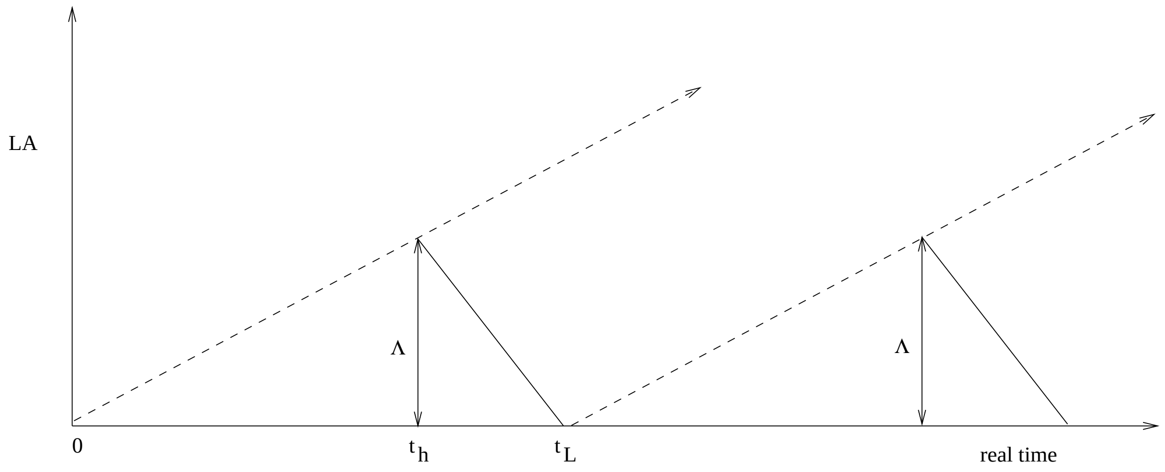


Figure 4.20: Lookahead with a Sliding Lookahead Window.

The probability of each of the events shown in Figures 4.17 through 4.19 is multiplied by the speedup for each event in order to derive the average speedup. For the case shown in Figure 4.17, the speedup (C_r) is provided by the time to read the cache over directly computing the result. For the remaining cases the speedup is $PR_{X,Y|X=E[X]}$ which has been defined as $\frac{LVT_{X,Y|X=E[X]}}{t}$ as shown in Equation 4.39. The analytical

$$\eta \equiv \qquad (4.39)$$
$$P_{cache\ X|X=E[X]} C_r + (P_{late\ X|X=E[X]} + P_{slow\ X|X=E[X]}) PR_{X,Y|X=E[X]}$$

results for speedup are graphed in Figure 4.21. A high probability of out-of-tolerance rollback in Figure 4.21 results in a speedup of less than one. Real messages are always processed when they arrive at an Logical Process. Thus, no matter how late Virtual Network Configuration results are, the system will continue to run near real time. However, when Virtual Network Configuration results are very late due to a high proportion of out-of-tolerance messages, the Virtual Network Configuration system is slower than real time because out-of-tolerance rollback overhead processing occurs. Anti-messages must be sent to correct other Logical Processs which have processed messages which have now been found to be out of tolerance from the current Logical Process. This causes the speedup to be less than one when the out-of-tolerance probability is high. Thus, $PR_{X,Y|X=E[X]}$ will be less than one for the "slow" shown in Figure 4.19.

4.4 Prediction Accuracy: α

Accuracy is the ability of the system to predict future events. A higher degree of accuracy will result in more "cache hits" of the predicted state cache information. Smaller tolerances should result in greater system accuracy, but this comes at the cost of a reduction in speedup.

Assume for simplicity that the effects of non-causality are negligible for the analysis in this section. The effects of causality are discussed in more detail in Section 4.7.2. An Logical Process may deviate from the real object it represents because either the Logical Process does not accurately represent the actual entity or because events out-

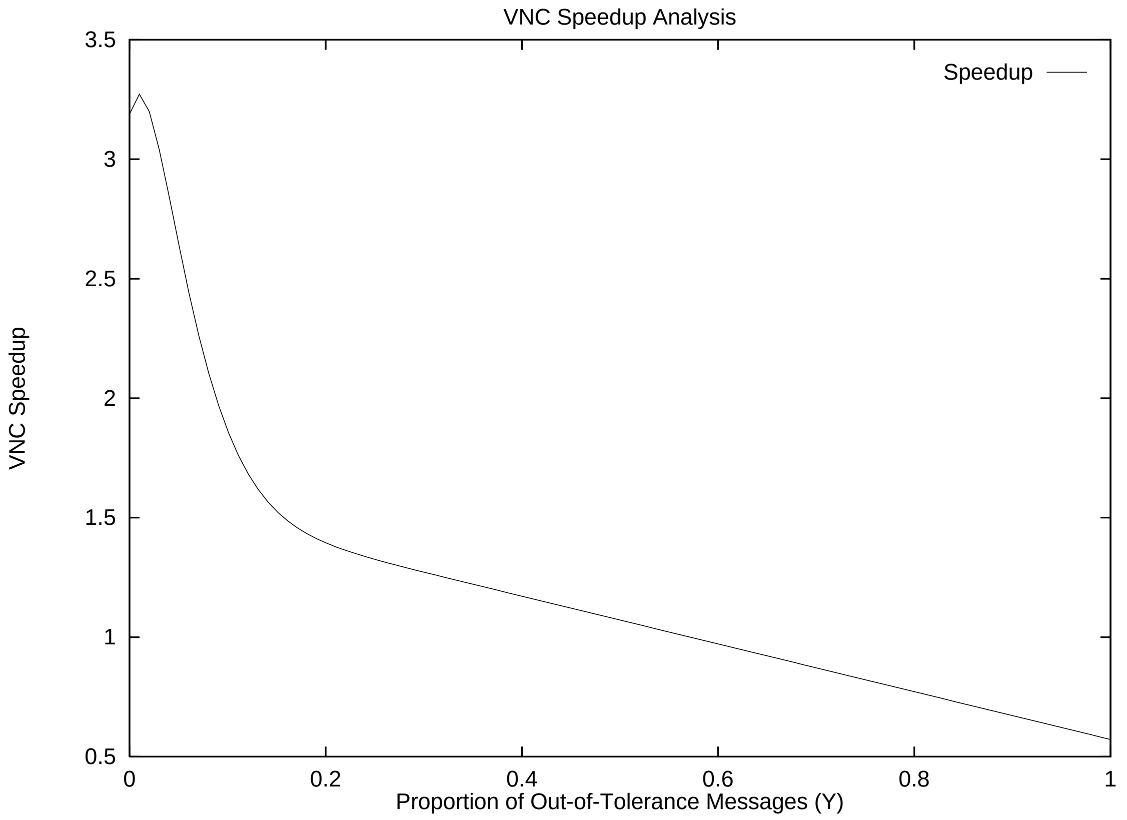


Figure 4.21: VNC Speedup.

side the scope of the predictive system may effect the entities being managed. Ignore events outside the scope of the predictive system for this analysis and consider only the deterministic error from inaccurate prediction of the driving process. The error is defined as the difference between an actual message value at the current time (v_t) and a message value which had been predicted earlier (v_{t_p}). Thus the **M**essage **E**rror is $ME = v_t - v_{t_p}$. Virtual message values generated from a driving process may contain some error. It is assumed that the error in any output message generated by a process is a function of any error in the input message and the amount of time it takes to process the message. A larger processing time increases the chances that external events may have changed before the processing has completed.

Two functions of total **AC**cumulated message value error ($AC(\cdot)$) in a predicted result are described by Equations 4.40 and 4.41 and are illustrated in Figure 4.22. ME_{lp_0} is the amount of error in the value of the virtual message injected into the predictive system by the driving process (lp_0). The error introduced into the value of the output message produced by the computation of each Logical Process is represented by the **C**omputation **E**rror function $CE_{lp_n}(ME_{lp_{n-1}}, t_{lp_n})$. The real time taken for the n^{th} Logical Process to generate a message is t_{lp_n}. The error accumulates in the State Queue (SQ) at each node by the amount $CE_{lp_n}(ME_{lp_{n-1}}, t_{lp_n})$ which is a function of the error contained in the input message from the predecessor Logical Process and the time to process that message. Figure 4.22 shows a driving process (DP) generating a virtual message which contains prediction error (ME_{lp_0}). The virtual message with prediction error (ME_{lp_0}) is processed by node LP_1 in t_{lp_1} time units resulting in an output message with error, $ME_{lp_1} = CE_{lp_0}(ME_{lp_0}, t_{lp_1})$.

Proposition 4 *The accumulated error in a message value is Equation 4.40 and Equation 4.41.*

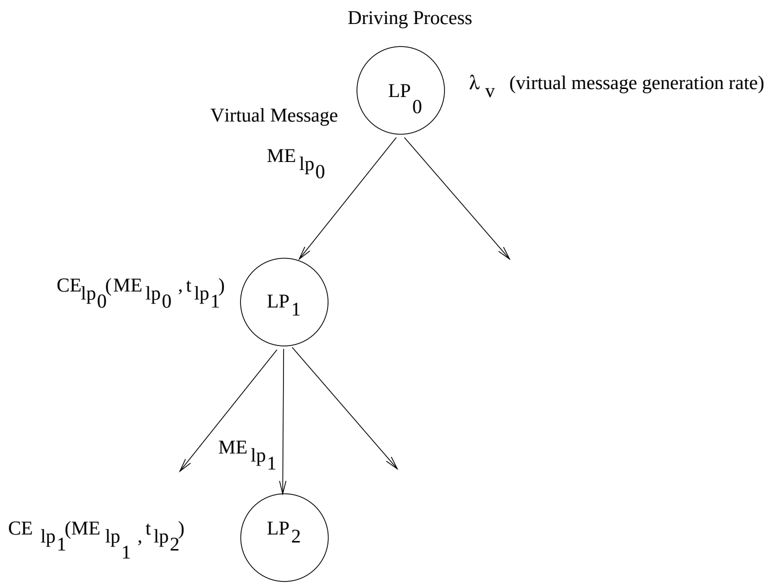

Driving Process
LP_0
λ_v (virtual message generation rate)
Virtual Message
ME_{lp_0}
$CE_{lp_0}(ME_{lp_0}, t_{lp_1})$
LP_1
ME_{lp_1}
$CE_{lp_1}(ME_{lp_1}, t_{lp_2})$
LP_2

Figure 4.22: Accumulated Message Value Error.

$$AC_n(n) = \sum_{i=1}^{N} CE_{lp_i}(ME_{lp_{i-1}}, t_{lp_i}) \tag{4.40}$$

$$AC_t(\tau) = \lim_{\sum t_{lp_i} \to \tau} \sum_{i=1}^{lpn} CE_{lp_i}(ME_{lp_{i-1}}, t_{lp_i}) \tag{4.41}$$

where CE_{lp_i} is the computational error added to a virtual output message value, ME_{lp_i} is the virtual message input error, and t_{lp_i} is the real time taken to process a virtual message.

As shown in Proposition 4, $AC_n(n)$ is the total accumulated error in the virtual message output by the n^{th} Logical Process from the driving process. $AC_t(\tau)$ is the accumulated error in τ real time units from the generation of the initial virtual message from the driving process. Equation 4.41 is $\lim_{\sum t_{lp_i} \to \tau} \sum_{i=1}^{lpn} AC_n(n)$, where lpn is the number of Logical Process computations in time τ. In other words, $AC_t(\tau)$ is the error accumulated as messages pass through lpn Logical Processs in real time τ.

For example, if a prediction result is generated in the third Logical Process from the driving process, then the total accumulated error in the result is $AC_n(3)$. If 10 represents the number of time units after the initial message was generated from the driving process then $AC_t(10)$ would be the amount of total accumulated error in the result. A cache hit occurs when $\mid AC_t(\tau) \mid \leqslant \Theta$, where Θ is the tolerance associated with the last Logical Process required to generate the final result. Equations 4.40 and 4.41 provide a means of representing the amount of error in a Virtual Network Configuration generated result. Once an event has been predicted and results pre-computed and cached, it would be useful to know what the probability is that the result has been accurately calculated, especially if any results are committed before a real message arrives. The out-of-tolerance check and rollback does not occur until a real message arrives. If a resource, such as a Virtual Circuit, is established ahead of time based on the predicted result, then this section has defined $\alpha = P[\mid AC_t(\Lambda) \mid > \Theta]$ where Θ is the tolerance associated with the last Logical Process required to generate the final result.

4.4.1 Rapidly Deployable Radio Network Error Accumulation

In the Rapidly Deployable Radio Network Network Control Protocol implementation, no network resources are committed, except for pre-loading the beam table, until the real message for an event is received. Assume the beamform table has an estimated download time of five milliseconds and a one degree tolerance for error in position. In the implementation used for the experimental validation, there is only one level of error propagation, and thus no error accumulation. Thus, α is shown in Equation 4.42 as illustrated in Figure 4.23 where d is the distance between the nodes, σ^2 is the variance in location prediction accuracy, and Θ is the tolerance. Angle e in Figure 4.23 is the variance of error in the beam steering angle due to the variance in the Cartesian co-

$$\alpha \equiv Pr\left[\tan^{-1}\left(\frac{\sigma}{d}\right) > \Theta\right] \tag{4.42}$$

ordinates provided by the prediction process.

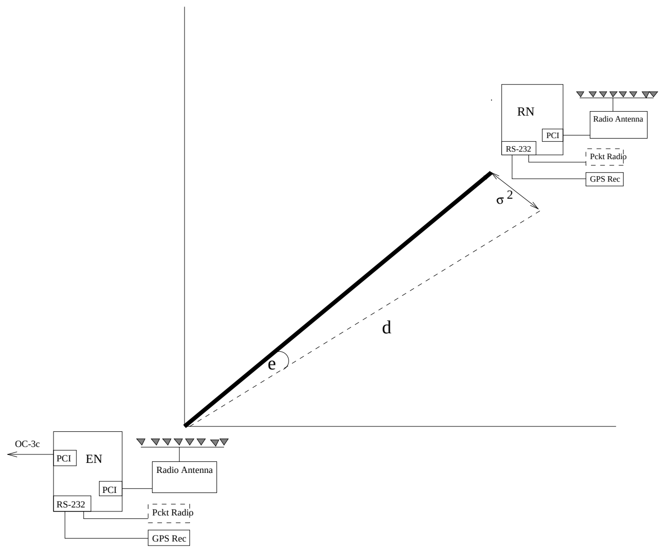


Figure 4.23: Beam Steering Variance.

4.5 Bandwidth: β

The amount of overhead in bandwidth required by Virtual Network Configuration is due to virtual and anti-message load. With perfect prediction capability, there should be exactly one virtual message from the driving process for each real message. The inter-rollback time, $\frac{1}{\lambda_{rb}}$, has been determined in Proposition 2 Equation 4.10. Virtual messages are arriving and generating new messages at a rate of λ_v. Thus, the worst case expected number of messages in the Send Queue which will be sent as anti-messages is $\frac{\lambda_v}{\lambda_{rb}}$ when a rollback occurs. The bandwidth overhead is shown in Equation 4.43 where λ_v is the virtual message load, λ_r is the real message load, and λ_{rb} is the expected rollback rate. The bandwidth overhead as a function of rollback rate is shown in Figure 4.24. Scalability in Virtual Network Configuration is the rate at which the proportion of rollbacks increases as the number of nodes increases. The graph in Figure 4.25 illustrates the tradeoff between the number of Logical Processs and the rollback rate given $\lambda_{vm} = 0.03$ virtual messages per millisecond, $\Delta_{vm} = 30.0$ milliseconds, $\tau_{task} = 7.0$ milliseconds, $\tau_{rb} = 1.0$ milliseconds, $S_{parallel} = 1.5$ and $C_r = 100$ where C_r is the speedup gained from reading the cache over computing the result. These parameters were chosen to accommodate the Edge Switch topology task which is currently the most time consuming Rapidly Deployable Radio Network operation and precludes the Edge Switch nodes from becoming mobile without Virtual Network Configuration and $Rm = \frac{2}{30\,\mathrm{ms}}$. The rollback rate in this graph is the sum of both the out-of-order and the out-of-tolerance rollback rates.

Proposition 5 *The expected bandwidth overhead is*

$$\beta = \frac{\frac{\lambda_v}{\lambda_{rb}} + \lambda_v + \lambda_r}{\lambda_r} \tag{4.43}$$

where λ_{rb} is the expected rollback rate, λ_v is the expected virtual message rate, and λ_r is the expected real message rate.

4.6 Performance Analysis

Equation 4.44 shows the complete Virtual Network Configuration performance utility. The surface plot showing the utility of Virtual Network Configuration as a function of the proportion of out-of-tolerance messages is shown in Figure 4.26 where Φ_s, Φ_w, Φ_b

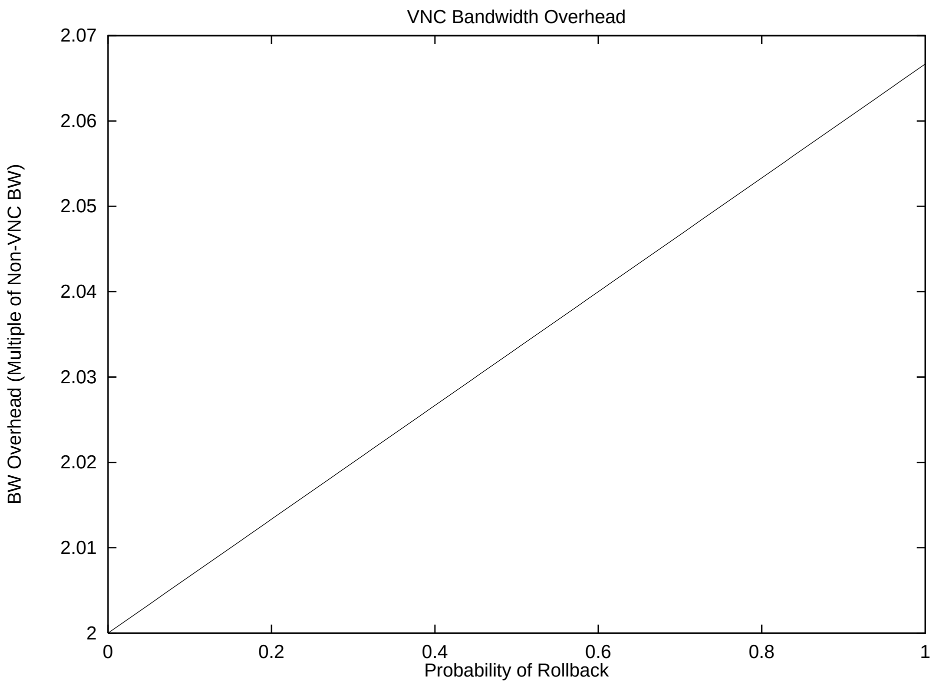


Figure 4.24: VNC Bandwidth Overhead.

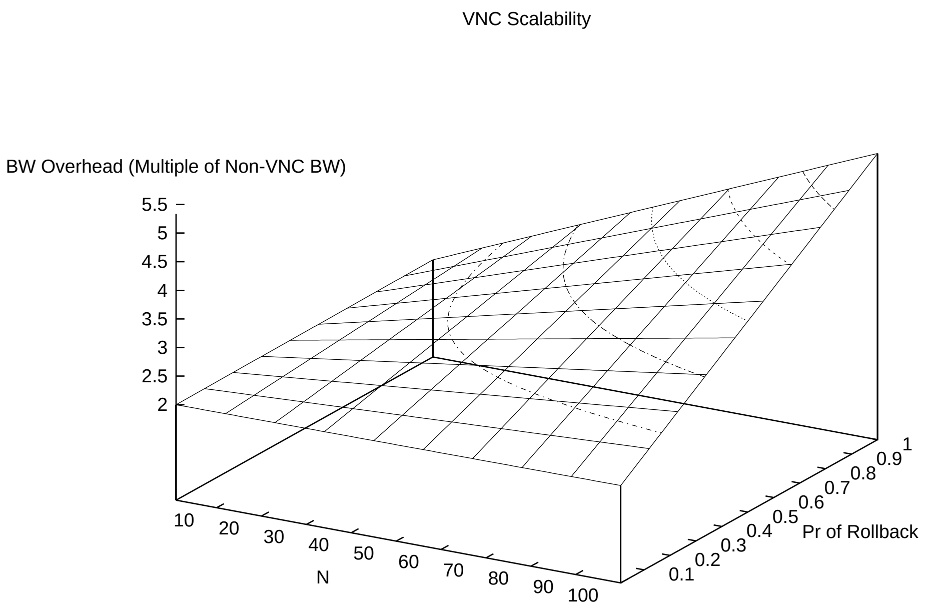


Figure 4.25: VNC Scalability.

are one and $\lambda_{vm} = 0.03$ virtual messages per millisecond, $\Delta_{vm} = 30.0$ milliseconds, $\tau_{task} = 7.0$ milliseconds, $\tau_{rb} = 1.0$ milliseconds, $S_{parallel} = 1.5$ and $C_r = 100$ where C_r is the speedup gained from reading the cache over computing the result. These parameters were chosen to accommodate the Edge Switch topology task which is currently the most time consuming Rapidly Deployable Radio Network operation and precludes the Edge Switch nodes from becoming mobile without Virtual Network Configuration. The wasted resources utility is not included in Figure 4.26 because as discussed in Section 4.4 there is only one level of message generation and thus no error accumulation. The y-axis is the relative marginal utility of speedup over reduction in bandwidth overhead $SB = \frac{\Phi_s}{\Phi_b}$. Thus if bandwidth reduction is much more important than speedup, the utility is low and the proportion of rollback messages would have to be kept below 0.3 per millisecond in this case. However, if speedup is the primary desire relative to bandwidth, the proportion of out-of-tolerance rollback message values can be as high as 0.5 per millisecond. If the proportion of out-of-tolerance messages becomes too high, the utility becomes negative because prediction time begins to fall behind real time.

The effect of the proportion of out-of-order and out-of-tolerance messages on Virtual Network Configuration speedup is shown in Figure 4.27. This graph shows that out-of-tolerance rollbacks have a greater impact on speedup than out-of-order rollbacks. The reason for the greater impact of the proportion of out-of-tolerance messages is that such rollbacks caused by such messages always cause a process to rollback to real time. An out-of-order rollback only requires the process to rollback to the previous saved state.

Figure 4.28 shows the effect of the proportion of virtual messages and expected lookahead per virtual message on speedup. This graph is interesting because it shows how the proportion of virtual messages injected into the Virtual Network Configuration system and the expected lookahead time of each message can affect the speedup. The

$$U_{VNC} = \Big(P_{cache\ X|X=E[X]} C_r + (P_{late\ X|X=E[X]} + P_{slow\ X|X=E[X]}) PR_{X,Y|X=E[X]}\Big) \Phi_s - P[\mid AC_t(\Lambda) \mid > \Theta]\Phi_w - \left(\frac{\frac{\lambda_v}{\lambda_{rb}} + \lambda_v}{\lambda_r}\right) \Phi_b \tag{4.44}$$

real and virtual message rates are $\frac{0.1}{\text{ms}}$, $Rm = \frac{2}{30\text{ ms}}$, $\lambda_{vm} = 0.03$ virtual messages per millisecond, $\Delta_{vm} = 30.0$ milliseconds, $\tau_{task} = 7.0$ milliseconds, $\tau_{rb} = 1.0$ milliseconds, $S_{parallel} = 1.5$ and $C_r = 100$ where C_r is the speedup gained from reading the cache over computing the result. These parameters were chosen to accommodate the Edge Switch topology task which is currently the most time consuming Rapidly Deployable Radio Network operation and precludes the Edge Switch nodes from becoming mobile without Virtual Network Configuration.

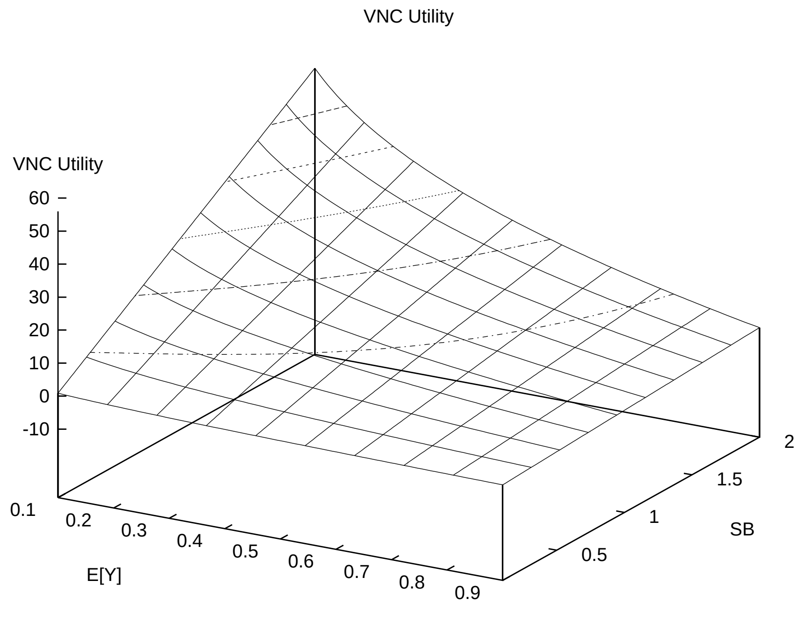


Figure 4.26: Overhead versus Speedup as a Function of Probability of Rollback.

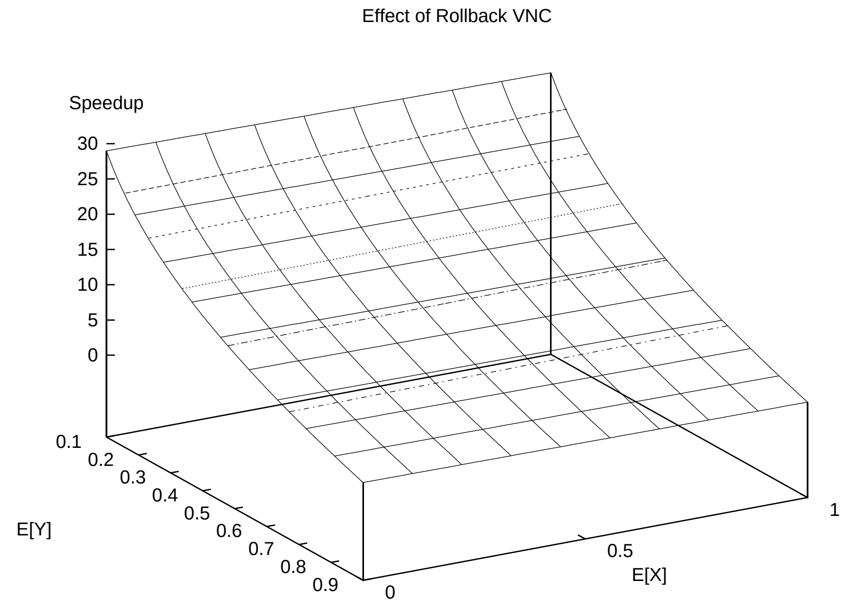


Figure 4.27: Effect of Non-Causality and Tolerance on Speedup.

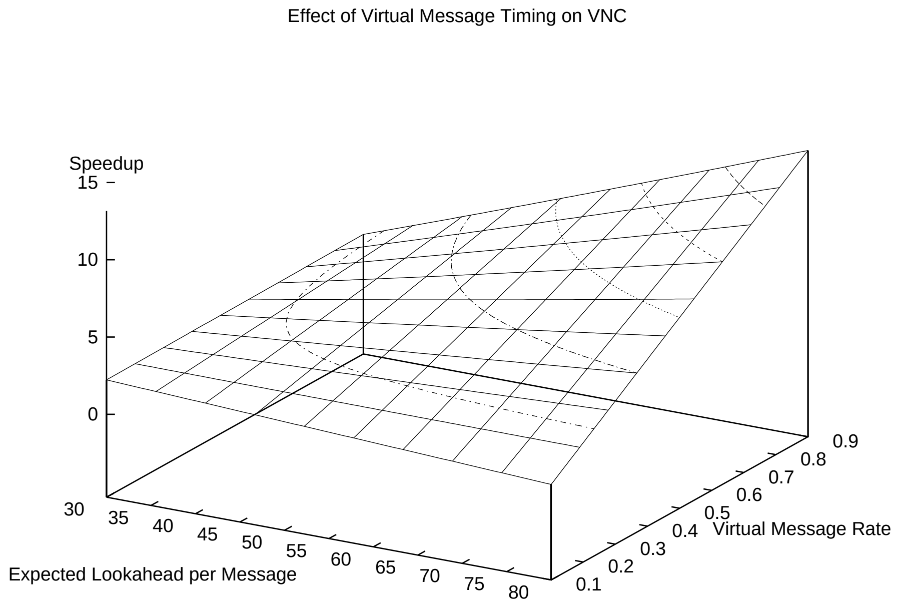


Figure 4.28: Effect of Virtual Message Rate and Lookahead on Speedup.

4.7 Aspects of Virtual Network Configuration Performance

The following sections discuss other aspects and optimizations of the Virtual Network Configuration algorithm including handling multiple future events and the relevance of Global Virtual Time (GVT) to Virtual Network Configuration. Since all possible alternative events cannot be predicted, only the most likely events are predicted in Virtual Network Configuration. However, knowledge of alternative events with a lower probability of occurrence will allow the system to prepare more intelligently.

Another consideration is the calculation of Global Virtual Time. This requires bandwidth and processing overhead. A discussion of the need for Global Virtual Time and an alternative to using Global Virtual Time which makes use of the accurate time provided by the Global Positioning System receiver is discussed in this section. Finally, a bandwidth optimization is suggested in which real packets may be sent less frequently.

4.7.1 Multiple Future Events

The architecture for implementing alternative futures discussed in Section3.1, while a simple and natural extension of the Virtual Network Configuration algorithm, creates additional messages and increases message sizes. Messages require an additional field to identify the probability of occurrence and an event identifier. However, the Virtual Network Configuration tolerance is shown to provide consideration of events which fall within the tolerances Θ_n where $n \in N$ and N is the number of Logical Processs.

The set of possible futures at time t is represented by the set E. A message value generating an event occurring in one of the possible futures is represented by E_{val}. As messages propagate through the Virtual Network Configuration system, there is a neighborhood around each message value defined by the tolerance (Θ_n). However, each message value also accumulates error ($AC_n(n)$). Let the neighborhood (E_Δ) be defined such that $E_\Delta \leqslant \mid \Theta_n - AC_n(n) \mid$ for each $n \in \{LogicalProcesss\}$. Thus, $\mid E_\Delta + AC_n(n) \mid \leqslant \min_{n \in N} \Theta_n$ defines a valid prediction. The infinite set of events in the neighborhood $E_\Delta \leqslant \mid \min_{n \in N} \Theta_n - AC_n(n) \mid$ are valid. Therefore, multiple future events which fall within the bounds of the tolerances reduced by any accumulated error can be implicitly considered.

4.7.2 Global Virtual Time

In order to maintain the lookahead (Λ), for the entire configuration system, it is necessary to know how far into the future the system is currently predicting. The purpose of Global Virtual Time (GVT) is to determine Λ where Λ is used to stop the Virtual Network Configuration system from looking ahead once the system has predicted up to the lookahead time. This helps maintain synchronization and saves processing and bandwidth since it is not necessary to continue the prediction process indefinitely into the future, especially since the prediction process is assumed to be less accurate the further it predicts into the future.

Distributed simulation mechanisms require Global Virtual Time in order to determine when to commit events. This is because the simulation cannot rollback beyond Global Virtual Time. In Virtual Network Configuration, event results are assumed to be cached before real time reaches the Local Virtual Time of an Logical Process. The only purpose for Global Virtual Time in Virtual Network Configuration is to act as a throttle on computation into the future. Thus, the complexity and overhead required to accurately determine the Global Virtual Time is unnecessary in Virtual Network Configuration. In the Virtual Network Configuration system implemented for Rapidly Deployable Radio Network, while the Local Virtual Time of an Logical Process is greater than $t + \Lambda$, the Logical Process does not process virtual messages. An accurate time t is provided by the Global Positioning System receiver. This has worked well in providing the necessary throttle for Logical Process processing. It is knowledge of the accurate Global Positioning System time that eliminates the need for Global Virtual Time.

4.7.3 Real Message Optimization

Real messages are only used in the Virtual Network Configuration algorithm as a verification that a prediction has been accurate within a given tolerance. The driving process need not send a real message if the virtual messages are within tolerance of the lowest tolerance within the path of a virtual message. This requires that the driving process have knowledge of the destination processes' tolerance. The driving process will have copies of previously sent messages in its send queue. If real messages are only sent when an out-of-tolerance condition occurs, then the orderwire load can be reduced by up to 50%. Figure 4.29 compares the orderwire load with and without the real message optimization.

4.8 Analysis Summary

The performance analysis of Virtual Network Configuration has quantified the costs versus the speedup provided by Virtual Network Configuration. The costs have been identified as the additional bandwidth and possible wasted resources due to inaccurate prediction. Since the Virtual Network Configuration algorithm combines optimistic synchronization with a real time system, the probability of non-causal message order was determined. A new approach using Petri-Nets and synchronic distance determined the likelihood of out-of-order virtual messages. This new Petri-Net approach was ap-

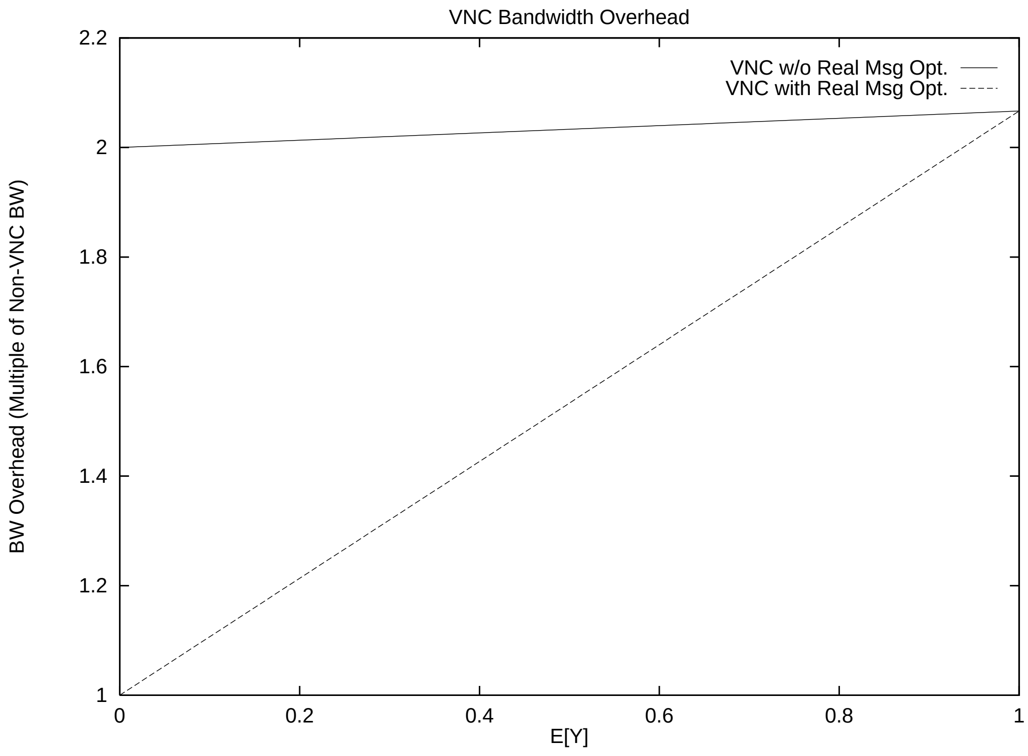


Figure 4.29: Bandwidth Overhead Reduction.

plied to the Rapidly Deployable Radio Network Network Control Protocol to determine the ability of Network Control Protocol to enforce causality in the virtual messages. Other factors such as the distribution of partitioned tasks into Logical Processs were discussed.

The speedup was defined as the expected rate of change of the Local Virtual Time with respect to real time. The speedup was quantified and a sensitivity analysis revealed the parameters most affecting speedup. The bandwidth was quantified based on the probability of rollback and the expected rollback rate of the Virtual Network Configuration system. A general analysis of the accumulated error of the Virtual Network Configuration system followed with the probability of error in the Rapidly Deployable Radio Network Network Control Protocol. Finally, the consideration of alternative future events, the relevance of Global Virtual Time, and a bandwidth optimization technique were discussed. The next section describes the experimental validation method and compares the results with the analysis in this section.

Chapter 5

Experimental Validation

5.1 Virtual Network Configuration Experimental Validation Overview

The experimental validation of Virtual Network Configuration examines simulations and actual implementations of the Virtual Network Configuration algorithm in order to determine the speedup (η) and bandwidth overhead (β) of the Virtual Network Configuration algorithm. The experimental validation is also an existence proof which demonstrates the feasibility of the Virtual Network Configuration concepts. For the Rapidly Deployable Radio Network, all cached results from the Virtual Network Configuration implementation are based directly on position and the amount of position error is a controlled variable, thus there is no accumulated error.

The implementation of the driving process is critical for Virtual Network Configuration, therefore this chapter begins with a discussion of an ideal implementation of the driving process for Rapidly Deployable Radio Network position prediction. This indicates the accuracy that can be expected from a position prediction process. However, the implementation of the driving process for the experimental validation in this chapter simulates Global Positioning System input. By simulating the Global Positioning System, the amount of error in virtual messages can be accurately controlled.

Virtual Network Configuration has been experimental validated in a mobile network configuration environment and in a simulated predictive network management environment. In the mobile network environment, a portion of the Rapidly Deployable Radio Network Network Control Protocol has been enhanced with Virtual Network Configuration. Specifically, the task which processes **USER_POS** messages pre-computes and caches beam tables. In the simulated predictive network management environment, a standards based management system enhanced with Virtual Network Configuration is simulated with Maisie [6].

5.2 The Driving Process

The Virtual Network Configuration driving process for mobile systems will require accurate position prediction. Previous mobile host location prediction algorithms have focused on an aggregate view of mobile host location prediction, primarily for such purposes as base-station channel assignment and base-station capacity planning. Examples are a fluid flow model [112] and the method of Hong and Rappaport [46]. A location prediction algorithm accurate enough for individual mobile host prediction has been developed in [72]. A brief overview of the algorithm follows because the algorithm in [72] is an ideal example of a driving process for Virtual Network Configuration and demonstrates the speedup that Virtual Network Configuration is capable of providing with this prediction method. The algorithm allows individual mobile hosts to predict their future movement based on past history and known constraints in the mobile host's path.

$$\{M(k,t)\} = \{S_{k,t} \mid k \leqslant K, t \in T\} + \{X(k,t) \mid k \leqslant K, t \in T\} \tag{5.1}$$

$$\{X(k,t)\} = \{M(k,t)\} - (\{M_c(k,t) \mid k \leqslant K, t \in T\} + \{M_t(k,t) \mid k \leqslant K, t \in T\}) \tag{5.2}$$

All movement ($\{M(k,t)\}$) is broken into two parts, regular and random motion. A Markov model is formed based on past history of regular and random motion and used to build a prediction mechanism for future movement as shown in Equation 5.1. The regular movement is identified by $S_{k,t}$ where S is the state (geographical cell area) identified by state index k at time t and the random movement is identified similarly by $X(k,t)$. $M(k,t)$ is the sum of the regular and random movement.

The mobile host location prediction algorithm in [72] determines regular movement as it occurs, then classifies and saves each regular move as part of a movement track or movement circle. A movement circle is a series of position states which lead back to the initial state while a movement track leads from one point to another distinct point. A movement circle can be composed of movement tracks. Let M_c denote a movement circle and M_t denote a movement track. Then Equation 5.2 shows the random portion of the movement.

The result of this algorithm is a constantly updating model of past movement classified into regular and random movement. The proportion of random movement to regular movement is called the randomness factor. Simulation of this mobility algorithm in [72] indicates a prediction efficiency of 95%. The prediction efficiency is defined as the prediction accuracy rate over the regularity factor. The prediction accuracy rate is defined in [72] as the probability of a correct prediction. The regularity factor is the proportion of regular states, $\{S_{k,t}\}$, to random states $\{X(k,t)\}$. The theoretically optimum line in [72, p. 143], may have been better labeled the deterministic line. The deterministic line is an upper bound on prediction performance for all *regular* movement. The addition of the random portion of the movement may increase or decrease actual prediction results above or below the deterministic line. A theoretically optimum (deterministic) prediction accuracy rate is one with a randomness factor of zero and a regularity factor of one. The algorithm in [72] does slightly worse than expected for completely deterministic regular movement but it improves as movement becomes more random. As a prediction algorithm for Virtual Network Configuration, a state as defined in [72] is chosen such that the area of the state corresponds exactly to the Virtual Network Configuration tolerance, then based on the prediction accuracy rate in the graph shown in [72, p. 143] the probability of being out of tolerance is less than 30% if the random movement ratio is kept below 0.4. An out-of-tolerance proportion of less than 30% where virtual messages are transmitted at a rate of $\lambda_{vm} = 0.03$ per millisecond results in a significant speedup as shown in Section 4.6.

In the Virtual Network Configuration architecture, each node could independently run this predictive location algorithm in order to predict its own future location and notify the edge switch via a virtual orderwire **USER_POS** network control message as shown in Table 2.5. In the current version of the orderwire, a much simpler linear prediction is used based on the last known location and speed. For testing the Virtual Network Configuration concept, it is only necessary to have a controlled amount of error for the predictive system. The same code used to simulate the next location is used to predict the next location with a controlled amount of error.

5.3 A Virtual Network Configuration Enhanced Rapidly Deployable Radio Network Orderwire Implementation

This section presents the results of performance measurements of the Virtual Network Configuration algorithm. The Virtual Network Configuration algorithm has been implemented for Rapidly Deployable Radio Network. Measurements have been taken in the environment shown in Figure 5.1. As mentioned previously, Global Positioning System (GPS) location is simulated such that Remote Nodes move in a linear path where the initial direction is chosen at random. The complete Network Control Protocol (NCP) code is used in this emulation environment except for the simulated Global Positioning System receiver which replaces the actual Global Positioning System driver interface code. In addition the beam table creation and download time is replaced with a delay which can be changed to examine the effect of task execution time on Virtual Network Configuration performance. The beam steering code includes beam steering optimization [40] between the Edge Switch and Remote Node nodes. Thus, most of the Network Control Protocol code is exercised in this emulation.

The details of the Network Control Protocol operation have been discussed in Section 2.5. A position update message (**USER_POS**) is sent by Remote Node nodes to Edge Switch nodes in order to update the Remote Node location. The first measurement is the time to process a position update message. This time begins from the receipt of an Remote Node position update message and ends when the Remote Node position update message has been processed. This may include the creation and download of a new beam table. When Virtual Network Configuration is disabled, the resulting beam creation times are shown in Figure 5.2. The y-axis in Figure 5.2 plots the time required

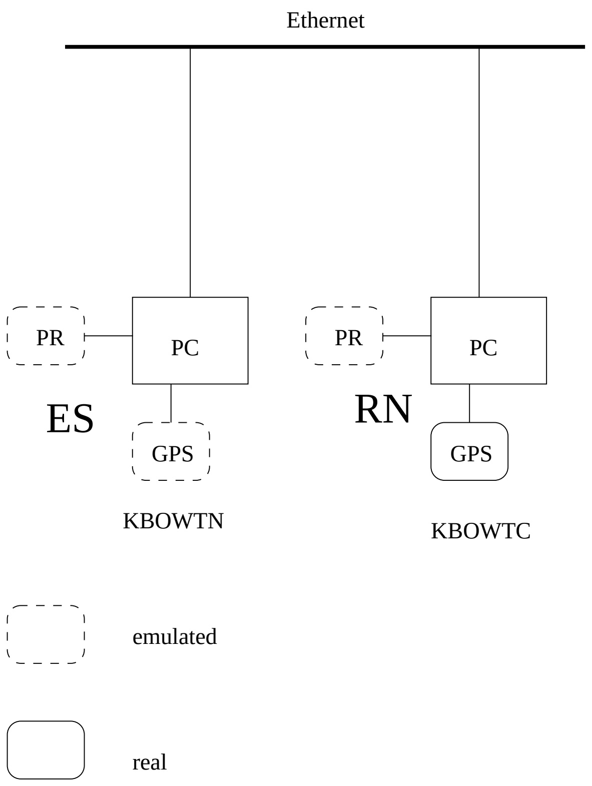


Figure 5.1: Virtual Network Configuration Test Environment.

to generate a beam table. The x-axis identifies a particular **USER_POS** message. There is not a significant variation in the beamform delay.

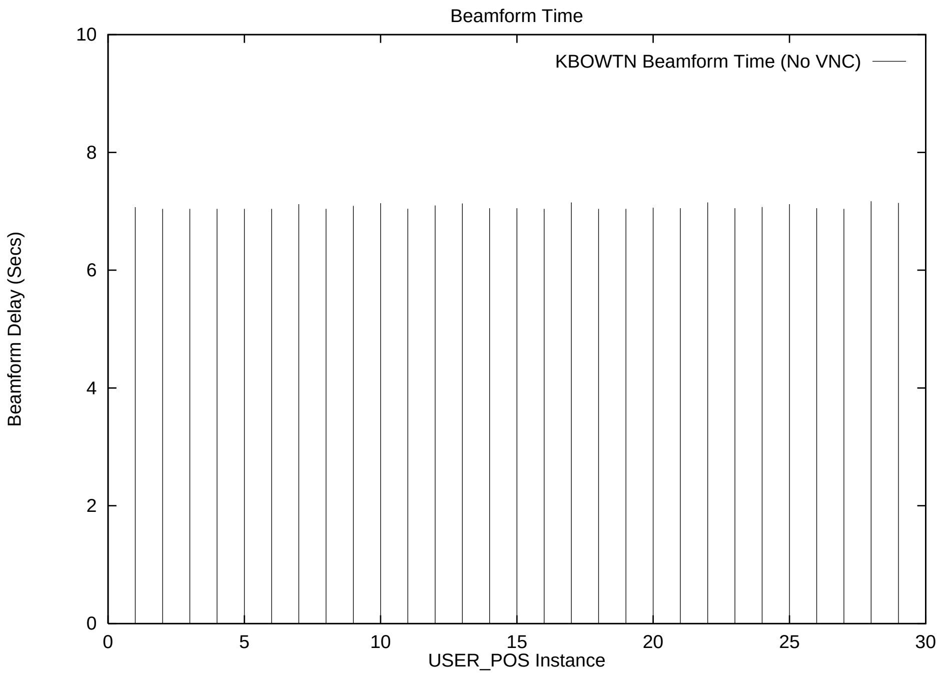


Figure 5.2: Edge Switch Beamforming Time without VNC.

The orderwire network load as a function of time as measured on the Edge Switch without Virtual Network Configuration is shown in Figure 5.3 and the orderwire load due to the Remote Node is shown in Figure 5.4. The load is normalized in terms of Data Encryption Standard encoded packets per second and is averaged over one second intervals. The traffic on the orderwire consists of the initial Edge Switch configuration followed by the Remote Node position update messages and handoff messages. The initial load shown in Figure 5.3 is caused by the Edge Switch configuration which quickly tapers off as Edge Switch configuration ends and **HANDOFF** messages may be transmitted.

The beam creation time is 3.5 times faster if the beam results have been cached, thus the speedup as determined from the analysis is shown in Figure 5.14. The ex-

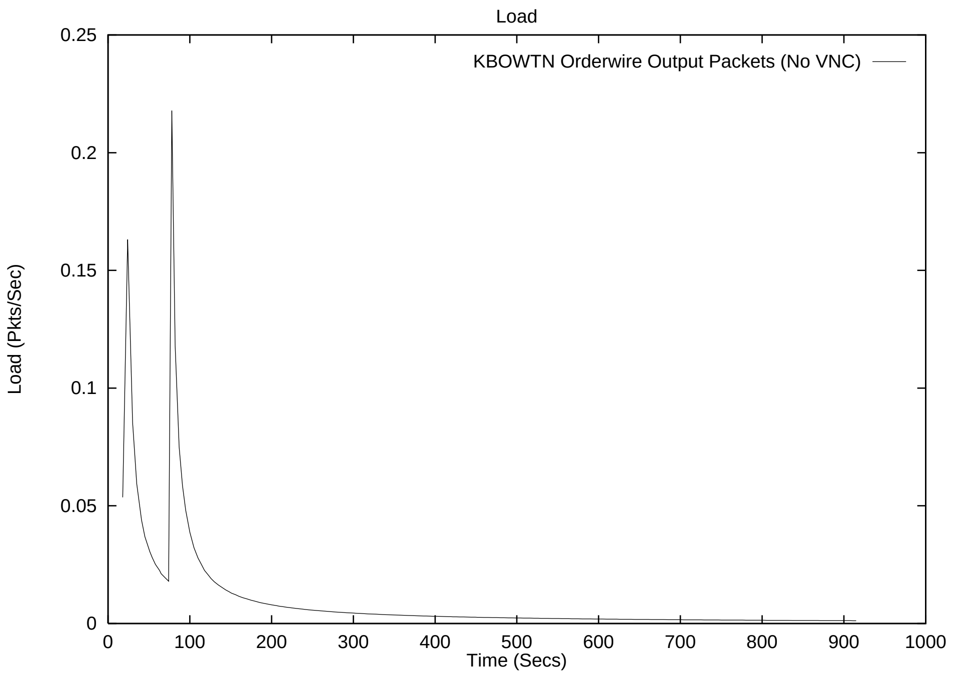


Figure 5.3: Edge Switch Orderwire Load without VNC.

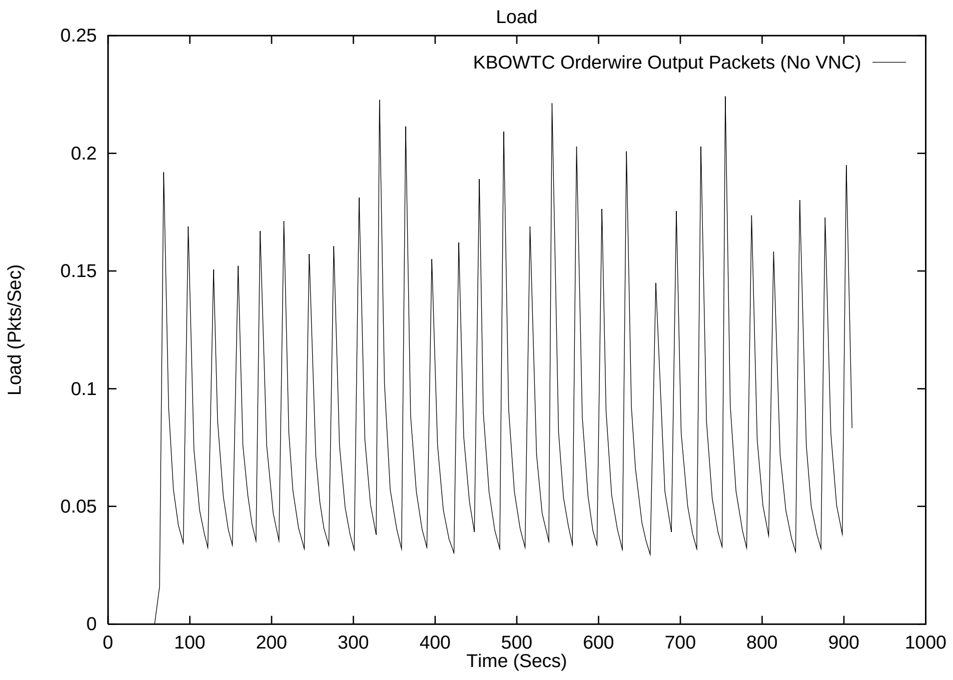


Figure 5.4: Remote Node Orderwire Load without VNC.

pected beam creation time without Virtual Network Configuration was 8.0 seconds. With perfect prediction, the expected beam creation time with Virtual Network Configuration was 2.29 seconds for a speedup of 3.5 times. The measured results are plotted with Figure 5.14 in Figure 5.14. The Virtual Network Configuration algorithm using perfect position prediction generated the beamform delay shown in Figure 5.5. The beam creation time has been significantly reduced. Figure 5.6 shows the Virtual Network Configuration orderwire traffic load from the Edge Switch and Figure 5.7 shows the load from the Remote Node. The load is approximately double the load shown in Figure 5.3.

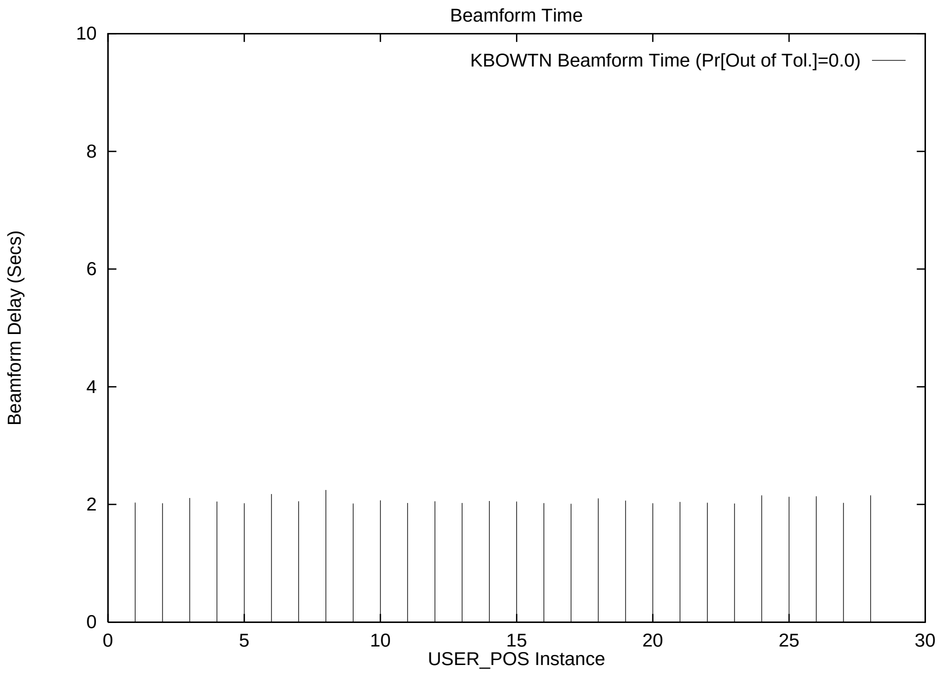


Figure 5.5: ES Beam formation Time with VNC and No Prediction Error.

Figure 5.8 shows the Remote Node Local Virtual Time as a function of time. This figure shows the Local Virtual Time quickly reaching the 30 second lookahead and delaying until the end of the lookahead window as expected. The prediction rate shown in Figure 5.8 is 1.23 virtual seconds per second as determined from the analysis in Section

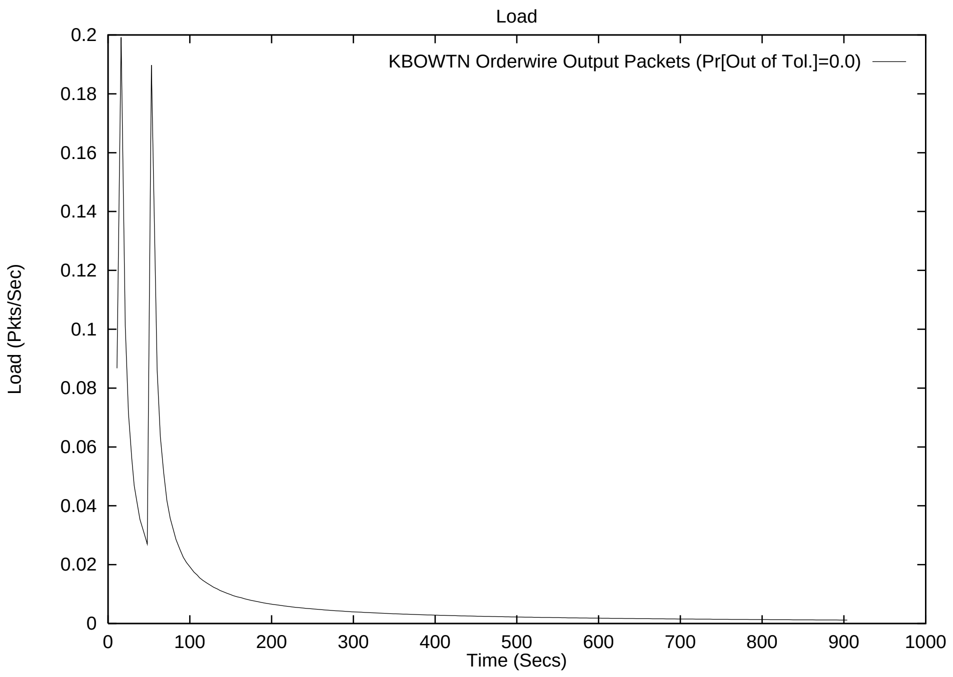


Figure 5.6: ES Orderwire Load with VNC and No Prediction Error.

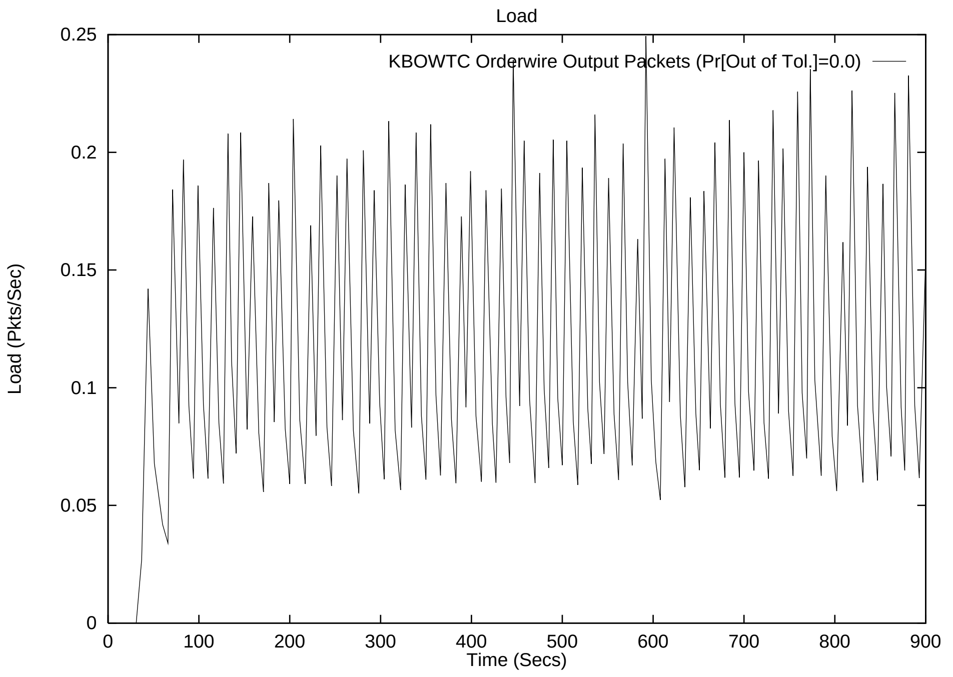


Figure 5.7: Remote Node Orderwire Load with VNC and No Prediction Error.

4.3.3 given $\lambda_{vm} = 0.03$ virtual messages per millisecond, $\Delta_{vm} = 30.0$ milliseconds, $\tau_{task} = 7.0$ milliseconds, $\tau_{rb} = 1.0$ milliseconds, $S_{parallel} = 1.5$ and $C_r = 100$ where C_r is the speedup gained from reading the cache over computing the result. These parameters were chosen to accommodate the Edge Switch topology task which is currently the most time consuming Rapidly Deployable Radio Network operation and precludes the Edge Switch nodes from becoming mobile without Virtual Network Configuration. Figure 5.9 shows the Edge Switch Local Virtual Time as a function of time. The Edge Switch Local Virtual Time is driven by the Receive Time (TR) of messages entering its input queue from the Remote Node. Thus the Edge Switch Local Virtual Time is similar to the Remote Node Local Virtual Time.

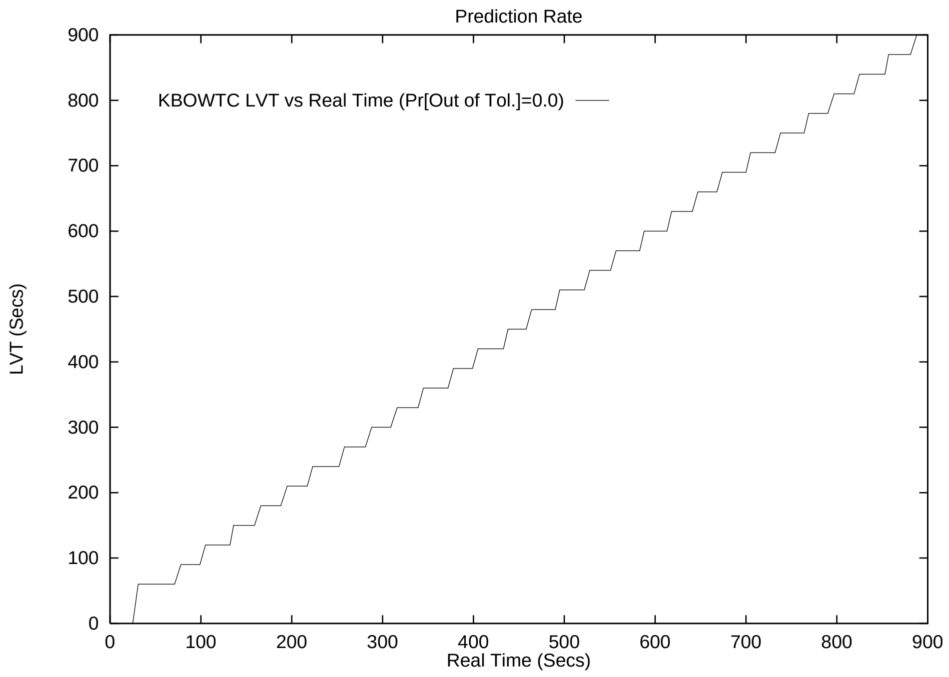


Figure 5.8: Remote Node LVT with VNC and No Prediction Error.

The next set of results show the performance of the Virtual Network Configuration algorithm as the prediction algorithm becomes less perfect. The predicted values are degraded in the driving process by adding an exponentially distributed error to the

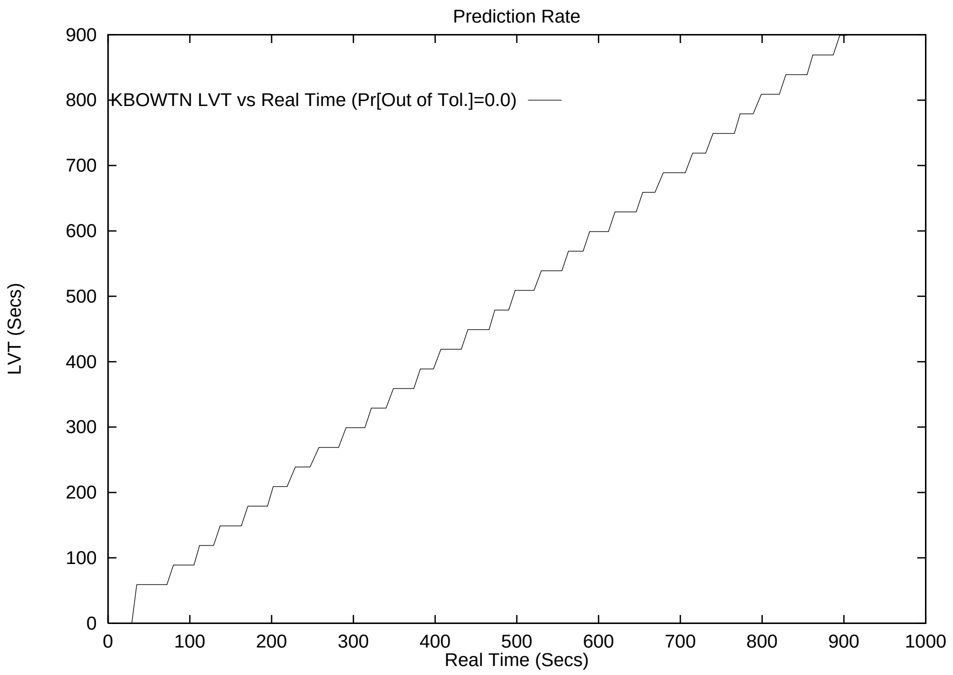


Figure 5.9: ES LVT with VNC and No Prediction Error.

predicted values. As discussed in the analysis section, rollback will increase the load on the orderwire and increase the user position message processing time. Figure 5.10 shows the Edge Switch orderwire load under the presence of rollback and Figure 5.11 shows the Remote Node load.

Compare the Edge Switch load without Virtual Network Configuration Figure 5.3, the Edge Switch load with Virtual Network Configuration and no prediction error in Figure 5.6, and the Edge Switch load with Virtual Network Configuration and large prediction error 5.10. The first peak in all the Edge Switch figures is the initial **MY-CALL** packet broadcast upon startup. The next peak is the initial **HANDOFF** packet sent from the Edge Switch to the Remote Node. Notice that the handoff occurs earlier in Figure 5.6 than it does in Figure 5.3. This due to the speedup gained via Virtual Network Configuration. However, in Figure 5.10, a third peak occurs which is a **HAND-OFF** anti-message. The **HANDOFF** anti-message is sent because the initial location prediction was out of tolerance. Once the handoff location has been corrected by the **HANDOFF** anti-message, no additional messages are sent from the Edge Switch.

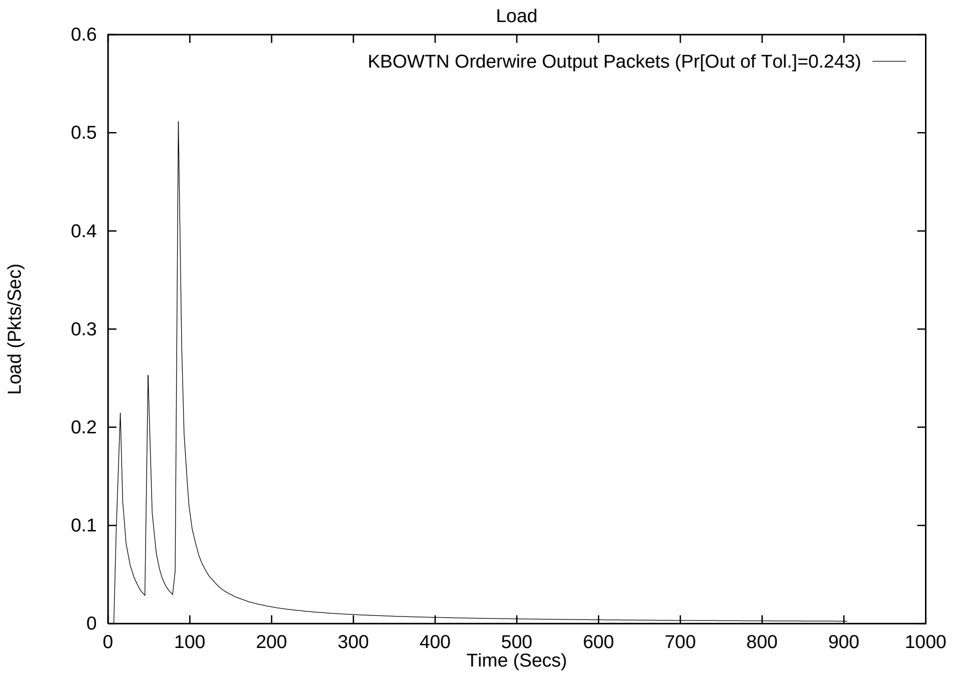


Figure 5.10: ES Orderwire Load with VNC and Large Prediction Error.

Figure 5.13 shows the analytical and measured bandwidth overhead as the prediction error increases. The bandwidth overhead increases by a small amount above twice the non-Virtual Network Configuration bandwidth as expected in the analysis. Figure 5.14 shows the analytical and measured speedup in the processing of a **USER_POS** packet as the prediction error increases. The measured results have a slightly lower speedup for low out-of-tolerance message probability and higher speedup than the analytical results for higher out-of-tolerance message probabilities. There are several possible explanations for the inaccuracy. First, the emulation was done on a shared processor with many other unrelated processes with different processor scheduling priorities. It is possible that these other processes have influenced the real time results. Second, the beam table creation task measured in this experimental implementation uses the

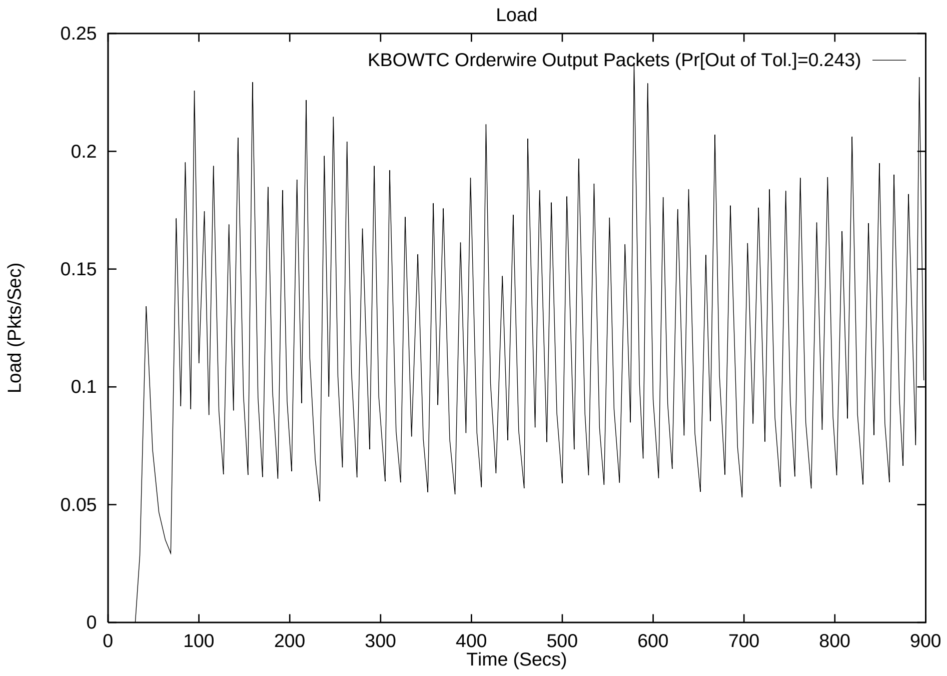


Figure 5.11: RN Orderwire Load with VNC and Large Prediction Error.

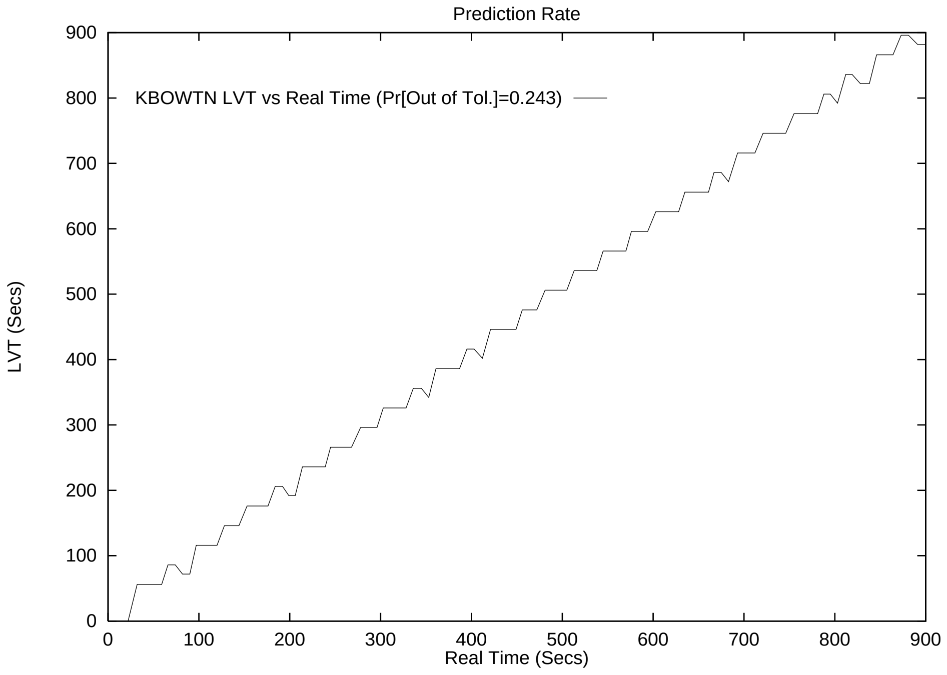


Figure 5.12: ES LVT with VNC and Large Prediction Error.

Virtual Network Configuration time warp mechanism across two different processors. The simplified speedup analysis for the contribution of time warp to Virtual Network Configuration speedup may not be precise enough. Finally, the analysis assumed an average time to perform the standard rollback (τ_{rb}) operations such as restoring the Logical Process state to a previously saved state from the State Queue (SQ) and modifying the Local Virtual Time. The actual time to perform a rollback may be smaller than the value used in the analysis. Over estimating τ_{rb} would have little or no effect when Y is small, but it would have a larger effect as Y increases which appears to be the case in Figure 5.14.

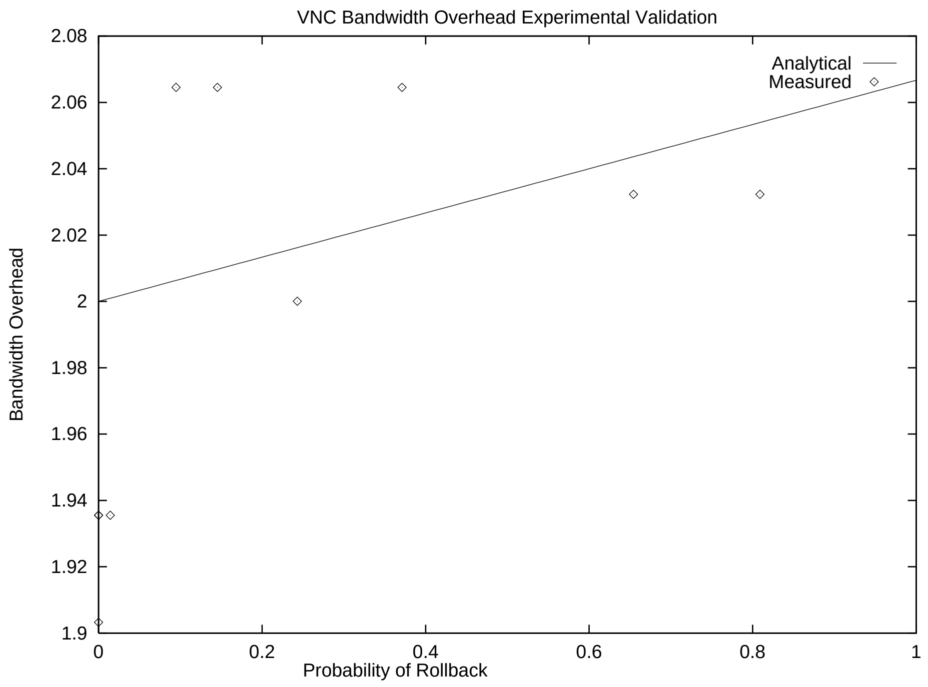


Figure 5.13: VNC Bandwidth Overhead.

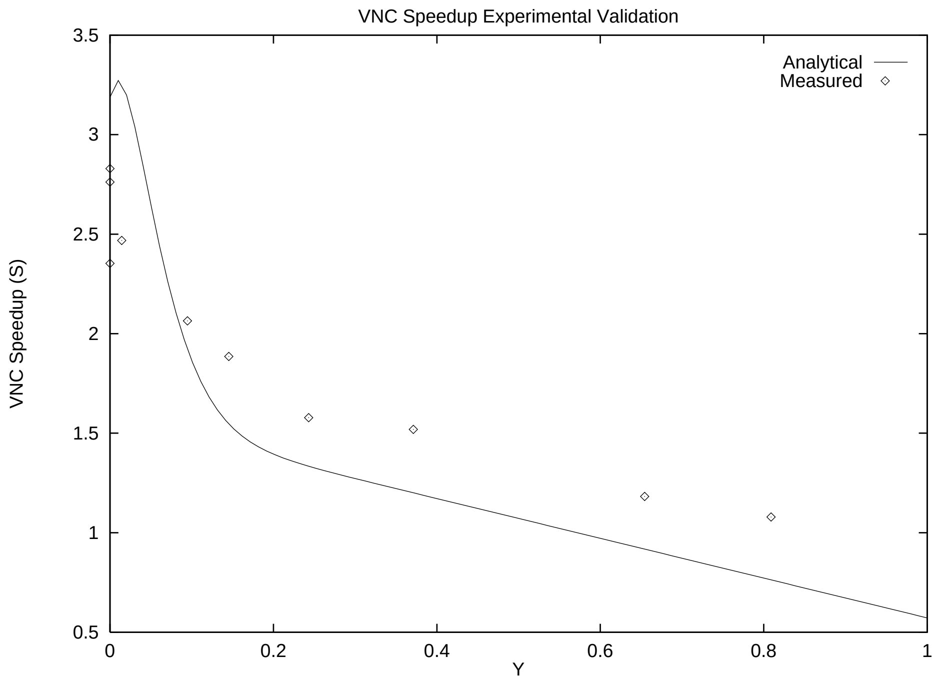


Figure 5.14: ES Beam Creation Speedup.

5.4 Virtual Network Configuration Network Control Protocol Experimental Implementation Summary

The Virtual Network Configuration algorithm is most sensitive to the proportion of messages which cause a rollback rate. The dominate cause of rollback is the proportion of messages which cause an out-of-tolerance rollback (Y) as determined in Section 4.3.4. Clearly minimizing Y has the greatest effect in optimizing the operation. However, if (Y) cannot be minimized, the virtual message generation rates and expected virtual message lookahead times can be adjusted as discussed in Section 4.6 and shown in Figure 4.28 in order to maximize the speedup.

The speedup provided by Virtual Network Configuration has been shown to be significant in the analysis and the experimental validation results. There are two contributing factors to the speedup in Virtual Network Configuration. The primary source of speedup is the computing ahead and caching of information required in the future. Another source of speedup in Virtual Network Configuration comes from the fact that optimistic advantage is taken of parallel processing available to the system.

The bandwidth overhead as a function of rollback rate and the number of Logical Processs is shown in Figure 4.25. This graph illustrates the trade-off between the number of Logical Processs and the rollback rate. The rollback rate in this graph is the sum of both the out-of-order and the out-of-tolerance rollback rates. The dominate contribution to the amount of overhead is the rollback rate and not the number of Logical Processs. The dominate overhead required by Virtual Network Configuration is additional bandwidth. A Virtual Network Configuration system which has perfectly adjusted parameters and perfect prediction will require twice the bandwidth of the same system without Virtual Network Configuration. As the rollback rate increases, more bandwidth is required. The rate of increase in bandwidth is shown in Figure 4.25.

The experimental implementation has shown the speedup of Virtual Network Configuration to decrease as a function of out-of-tolerance messages at the rate determined by the analysis. This is shown in Figure 5.14. Also, the bandwidth overhead has been shown to increase at a very slight rate as a function of the proportion of out-of-tolerance messages as shown in Figure 4.24.

In the next section, the Virtual Network Configuration algorithm is used to enhance the operation of a predictive network management application. The predictive capability of Virtual Network Configuration is utilized in the predictive management application, however, the overhead measurement is different as discussed in Section 5.5.

5.5 Predictive Network Management Simulation Results

While the goal of Virtual Network Configuration in the predictive management environment is to enable prediction just as in the Rapidly Deployable Radio Network environment, the focus on the overhead is different. As discussed later in this section, the goal in the predictive management system is to reduce the verification query overhead. The predictive management system is assumed to run on a separate processing system, thus the message overhead among Logical Processs, is not as critical as in Rapidly Deployable Radio Network. In the Rapidly Deployable Radio Network implementation, there is no verification query and the inter-Logical Process message is crucial since the transmission medium between Logical Processs is a low bandwidth packet radio system. Another measurement of interest for the Virtual Network Configuration enhanced predictive management environment is the prediction rate and the effects of out of tolerance and out of order rollback. Therefore, the metrics that this section focuses on are verification query overhead and prediction rate.

Systems management means the management of heterogeneous subsystems of network devices, processing platforms, distributed applications, and other components found in communications and computing environments. Current system management relies on presenting a model to the user of the managed system which should accurately reflect the current state of the system and should ideally be capable of predicting the future health of the system. System management relies on a combination of asynchronously generated alerts and polling to determine the health of a system [105].

The management application presents state information such as link state, buffer fill and packet loss to the user in the form of a model [81]. The model can be as simple as a passive display of nodes on a screen or a more active model which allows displayed nodes to change color based on state changes, or react to user input by allowing the user to manipulate the nodes which causes values to be set on the managed entity. This model can be made even more active by enhancing it with predictive capability [14]. This enables the management system to manage itself, for example, to optimize its polling rate. The two major management protocols, Simple Network Management Protocol [94] and Common Management Information Protocol [50], allow the management station to poll a managed entity to determine its state. In order to accomplish real-time and predictive network management in an efficient manner, the model should be updated with real-time state information when it becomes available, while other parts of the model work ahead in time. Those objects working ahead of real-time can predict future operation so that system management parameters such as polling times and thresholds can be dynamically adjusted and problems can be anticipated. The model will not deviate too far from reality because those processes which are found to deviate beyond a certain threshold will be rolled back, as explained in detail later.

One goal of this research is to minimize polling overhead in the management of large systems [109]. Instead of basing the polling rate on the characteristics of the data itself, the entity is emulated some time into the future in order to determine the characteristics of the data to be polled. Polling is still required with this predictive network management system in order to verify the accuracy of the emulation.

5.6 Predictive Standards-Based Network Management Information

Management information from standards-based managed entities must be mapped into this predictive network management system. Network management systems rely upon standard mechanisms to obtain the state of their managed entities in near real-time.

These mechanisms, SNMP [94] and CMIP [50] for example, use both solicited and unsolicited methods. The unsolicited method uses messages sent from a managed entity to the manager. These unsolicited messages are called traps or notifications; the former are not acknowledged while the latter are acknowledged. These messages are very similar to messages used in distributed simulation algorithms; they contain a timestamp and a value, they are sent to a particular destination, that is, a management entity, and they are the result of an event which has occurred.

Information requested by the management system from a particular managed entity is solicited information. It also corresponds to messages in distributed simulation. It provides a time and a value; however, not all such messages are equivalent to messages in distributed simulation and required in a predictive management system. These messages provide the management station with the current state of the managed entity, even though no change of state may have occurred or multiple state changes may have occurred. The design of a management system which requests information on the state of its managed entities at the optimum time has always been a problem in network management. If requested too frequently, bandwidth is wasted, if not requested frequently enough, critical state change information will be missed.

We will assume for simplicity that each managed entity is represented in the predictive management system by a Logical Process. It would greatly facilitate system management if vendors provide not only the standards based SNMP Management Information Base (MIB) as they do now, but also a standard simulation code which models the entity or application behavior and can be plugged into the management system just as in the case with a MIB. Vendors should have models of their devices readily available from product development.

5.7 Characteristics of the Predictive Network Management System

There are two types of false messages generated in this predictive network management system; those produced by messages arriving in the past Local Virtual Time (LVT) of an Logical Process and those produced because the Logical Process is generating results which do not match reality. If rollbacks occur for both reasons the question arises as to whether the system will be stable. An unstable system is one in which there exists enough rollbacks to cause the system to take longer than real-time to reach the end of the Sliding Lookahead Window (SLW). A stable system is able to make reasonably accurate predictions far enough into the future to be useful. An unstable system will have its performance degraded by rollbacks to the point where it is not able to predict ahead of real-time. Initial results shown later indicate that predictive network management systems can be stable.

There are several parameters in this predictive network management system which must be determined. The first is how often the predictive network management system should check the Logical Process to verify that past results match reality. There are two conditions which cause Logical Processs in the system to have states which differ from the system being managed and to produce inaccurate predictions. The first is that the predictive model which comprises an Logical Process is most likely a simplification of the actual managed entity and thus cannot model the entity with perfect fidelity. The second reason is that events outside the scope of the model may occur which lead to inaccurate results. However, a benefit of this system is that it will self-adjust for both of these conditions.

The optimum choice of verification query time, T_{query}, is important because querying entities is something the predictive management system should minimize while still guaranteeing that the accuracy is maintained within some predefined tolerance (Θ). For example, the network management station may predict user location as explained later. If the physical layer attempts spatial reuse via antenna beamforming techniques as in the RDRN project, then there is an acceptable amount of error in the steering angle for the beam and thus the node location is allowed a tolerance. The tolerances could be set for each state variable or message value sent from a Logical Process. State verification can be done in one of at least two ways. The Logical Process state can be compared with previously saved states as real time catches up to the saved state times or output message values can be compared with previously saved output messages in the send queue. In the prototype implemented for this predictive network management system state verification is done based on states saved in the state queue. This implies that all Logical Process states must be saved from the Logical Process LVT back to the current time.

The amount of time into the future which the emulation will attempt to venture is another parameter which must be determined. This Sliding Lookahead Window width (Λ) should be preconfigured based on the accuracy required; the farther ahead this predictive network management system attempts to go past real time, the more risk that is assumed.

5.7.1 Optimum Choice of Verification Query Times

As previously stated, the prototype system performs the verification based on the states in the state queue.

One method of choosing the verification query time would be to query the entity based on the frequency of the data we are trying to monitor. Assuming the simulated data is correct, query or sample in such a way as to perfectly reconstruct the data, for example, based on the maximum frequency component of the monitored data. A possible drawback is that the actual data may be changing at a multiple of the predicted rate. The samples may appear to to be accurate when they are invalid.

5.7.2 Verification Tolerance

The verification tolerance (Θ) is the amount of difference allowed between the Logical Process state and the actual entity state. A large tolerance decreases the number of false messages and rollbacks, thus increasing performance and requiring fewer queries, but allows a larger probability of error between predicted and the actual values will cause rollbacks in each Logical Process at real times of t_{vfail} from the start of execution of each Logical Process.

The error throughout the simulated system may be randomized in such a way that errors among Logical Processs cancel. However, if the simulation is composed of many of the same class of Logical Process, the errors may compound rather than cancel each other. The tolerance of a particular Logical Process (Θ_{lp_n}) will be reached in time $t_{vfail_n} = \{\text{lub } \tau \text{ s.t. } AC_t(\tau) > \Theta_{lp_n}\}$. The verification query period (Υ) should be less than or equal to t_{vfail_n} in order to maintain accuracy within the tolerance.

The accuracy of any predicted event must be quantified. This could be quantified as the probability of occurrence of a predicted event. The probability of occurrence will be a function of the verification tolerance, the time of last rollback due to verification error, the error between the simulation and actual entity, and the Sliding Lookahead Window. Every Logical Process will be in exact alignment with its Physical Process as a result of a state verification query. This occurs every $T_{query} = t_{vfail}$ time units.

5.7.3 Length of Lookahead Window

The length of the lookahead window (Λ) should be as large as possible while maintaining the required accuracy. The total error is also a function of the chain of messages which lead to the state in question. Thus the farther ahead of real-time the predictive network management system advances, $t_{ahead} = GVT - t_{current-time}$, the greater the number of messages before a verification query can be made and the greater the error. The maximum error is $AC_t(\Lambda)$.

5.7.4 Calibration Mode of Operation

It may be helpful to run the predictive network management system in a mode such that error between the actual entities and the predictive network management system are measured. This error information can be used during the normal predictive mode in order to help set the above parameters. This begins to remind one of back propagation in a neural network, that is, the predictive network management system automatically adjusts parameters in response to real output in order to become more accurate. This calibration mode could be part of normal operation. The error can be tracked simply by keeping track of the difference between the simulated messages and the result of verification queries.

5.8 Model and Simulation

A simulation of Virtual Network Configuration applied to a predictive management system was implemented with Maisie [6]. Maisie is the simulation environment used here. Its suitability for this has been demonstrated in the RDRN network management and control design and development and in [100] to develop a mobile wireless network parallel simulation environment.

5.8.1 Verification Query Rollback Versus Causality Rollback

Verification query rollbacks are the most critical part of the predictive management system. They are handled in a slightly different fashion from causality failure rollbacks. A state verification failure causes the Logical Process state to be corrected at the time of the state verification which failed. The state, S_v, has been obtained from the actual device from the verification query at time t_v. The Logical Process rolls back to exactly t_v with state, S_v. States greater than t_v are removed from the state queue. Anti-messages are sent from the output message queue for all messages greater than t_v. The Logical Process continues forward execution from this point. Note that this implies that the message and state queues cannot be purged of elements which are older than the GVT. Only elements which are older than real time can be purged.

5.8.2 The Predictive Management System Simulation

A small closed queuing network with FCFS servers is used as the target system in this study. Figure 5.15 shows the real system to be managed and the predictive management model. In this initial feasibility study, the managed system and the predictive management model are both modeled with Maisie. The verification query between the real system and the management model are explicitly illustrated in Figure 5.15.

The system consists of three switch-like entities, each switch contains a single queue and switches consisting of 10 exponentially distributed servers which must sequentially service each packet. A mean service time of 10 time units is assumed. The servers represent the link rate. The packet is then forwarded with equal probability to another switch, including itself. Each switch is a driving process; the switches forward

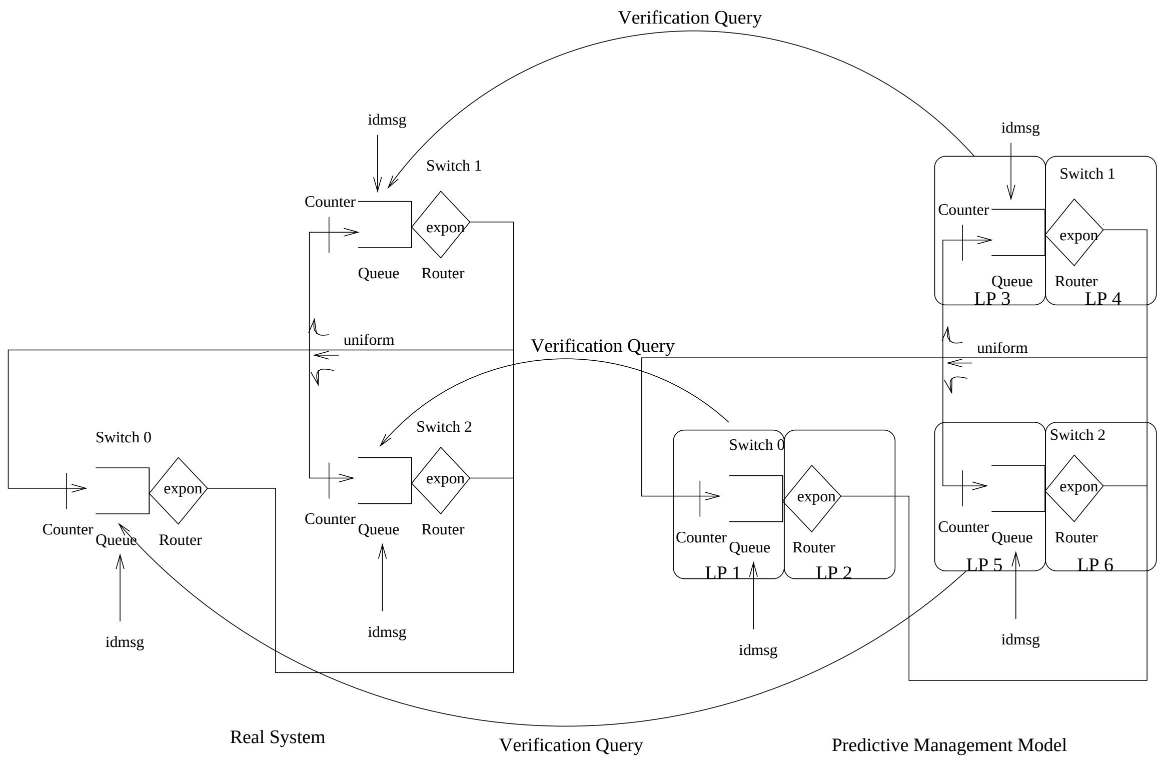


Figure 5.15: Initial Feasibility Network Model

real and virtual messages. The cumulative number of packets which have entered each switch and queue is the state. This is similar to SNMP [94] statistics monitored by SNMP Counters, for example, the **ifInOctets** counter in MIB-II interfaces [79].

Both real and virtual messages contain the time at which service ends. The switches are fully connected. An initial message enters each queue upon startup to associate a queue with its switch. This is the purpose of the **idmsg** which enters the queues in Figure 5.15. The predictive system parameters are more compactly identified as a triple consisting of Lookahead Window Size (seconds), Tolerance (counter value), and Verification Query Period (seconds) in the form $(\Lambda, \Theta, \Upsilon)$. The effect of these parameters are examined on the system of switches previously described. The simulation was run with the following triples: $(5, 10, 5)$, $(5, 10, 1)$, $(5, 3, 5)$, $(400, 5, 5)$. The graphs which follow show the results for each triple.

The first run parameters were $(5, 10, 5)$. There were no state verification rollbacks although there were some causality induced rollbacks as shown in Figure 5.16. GVT increased almost instantaneously versus real time; at times the next event far exceeded the look-ahead window. This is the reason for the nearly vertical jumps in the GVT as a function of real-time graph as shown in Figure 5.16. The state graph for this run is shown in Figure 5.17.

In the initial implementation, state verification was performed in the Logical Process immediately after each new message was received. However, the probability that an Logical Process had saved a future state, while processing at its LVT, with the same state save time as the time at which a real message arrived was low. Thus, there was frequently nothing with which to compare the current state in order to perform the state verification. However, it was observed that the predictive system was simulating up to the lookahead window very quickly and spending most of its time holding, during which time it was doing nothing. The implementation was modified so that each entity

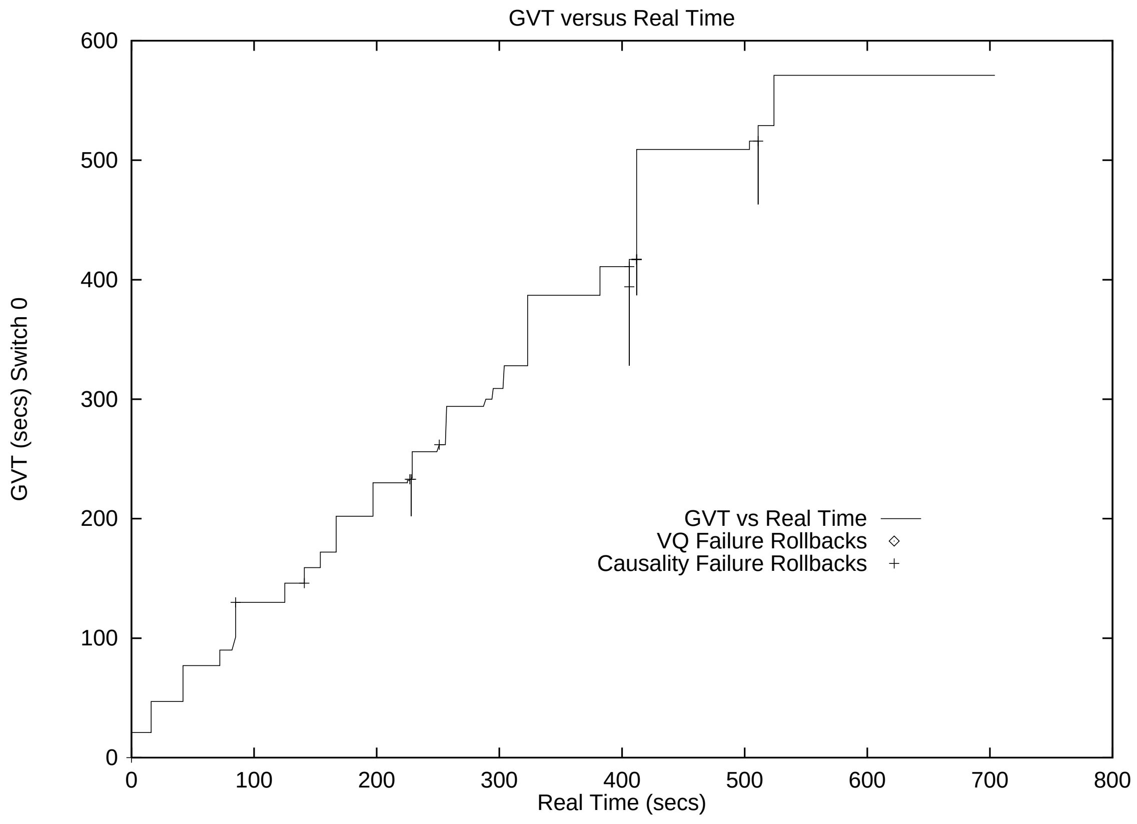


Figure 5.16: Rollbacks Due to State Verification Failure (5, 10, 5)

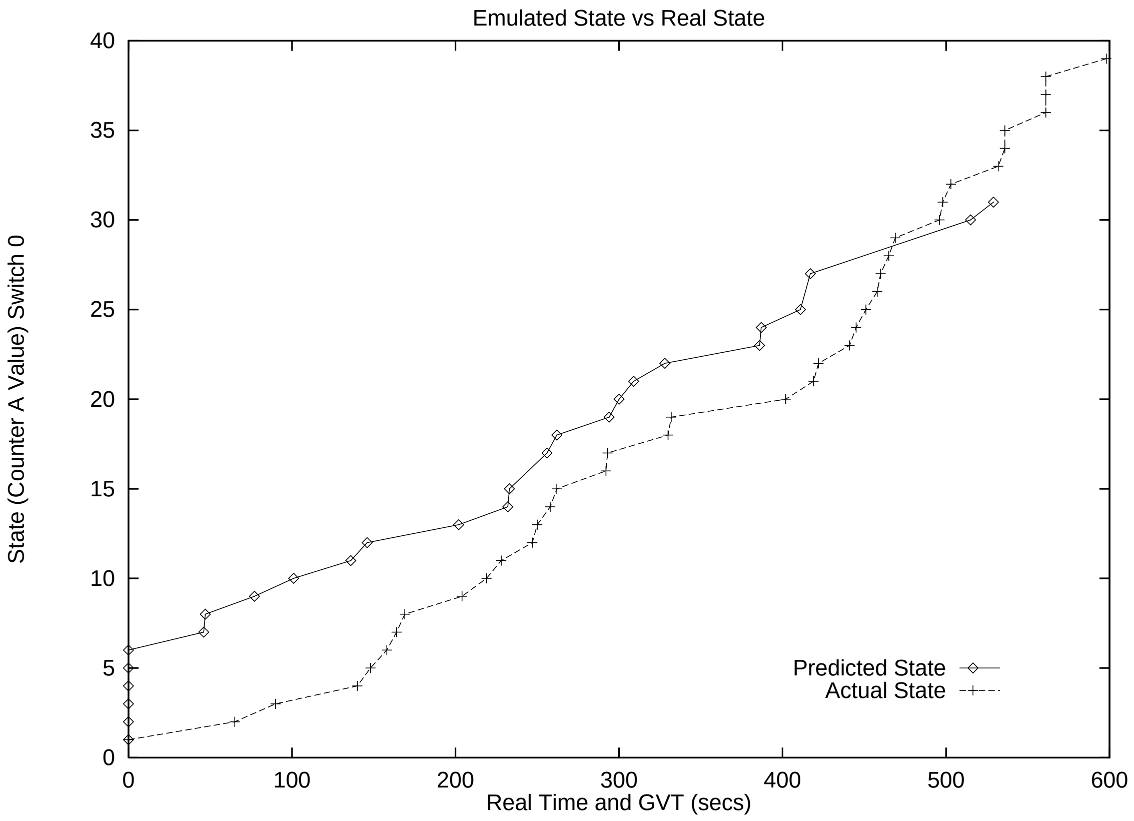


Figure 5.17: State (5, 10, 5)

would perform state verification during its hold time. This design change better utilized the processors and resulted in more accurate alignment between the actual and logical processes.

The results for the $(5, 10, 1)$ run were similar, except that the predictive and actual system comparisons were more frequent because the state verification period had been changed from once every 5 seconds to once every second. Error was measured as the difference in the predicted Logical Process state versus the actual system state. This run showed errors that were greater than those in the first run, great enough to cause state verification rollbacks. The error levels for both runs are shown in Figures 5.18 and 5.19. The state graph for this run is shown in Figure 5.20.

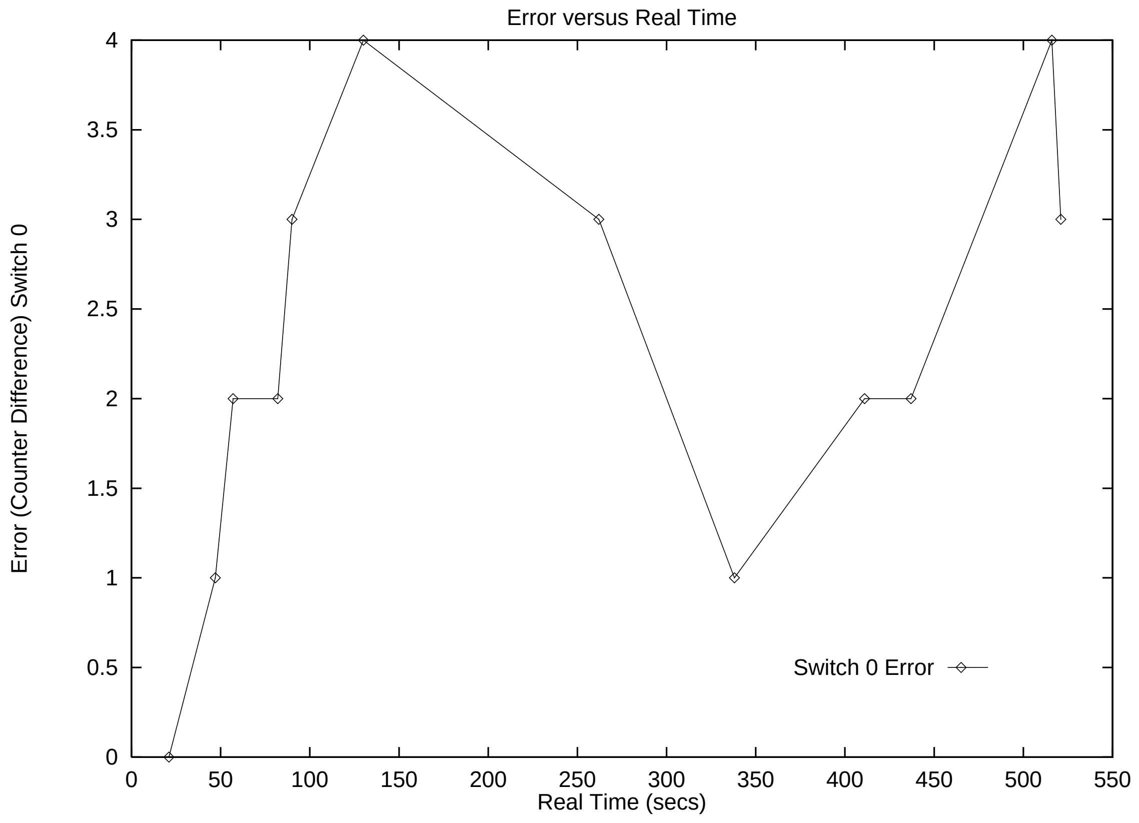


Figure 5.18: Amount of Error (5, 10, 5)

The next run used $(5, 3, 5)$ parameters. Here we see many more state verification

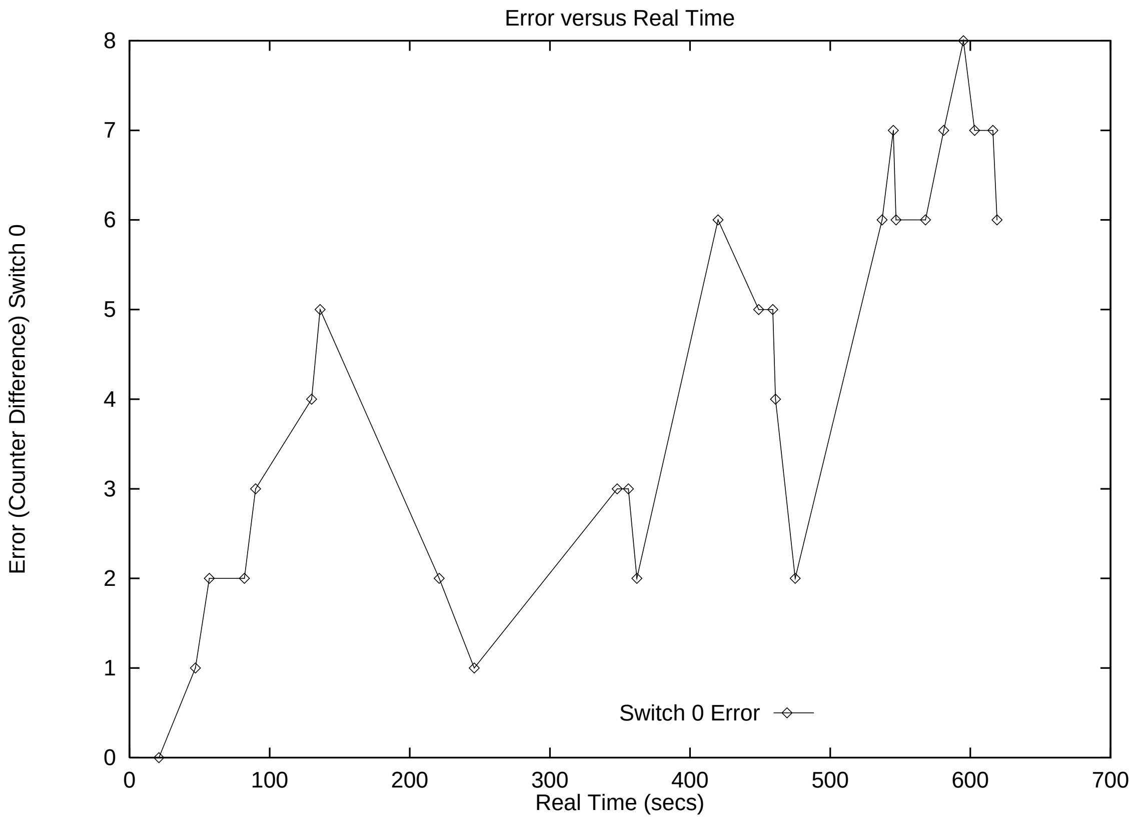


Figure 5.19: Amount of Error (5, 10, 1)

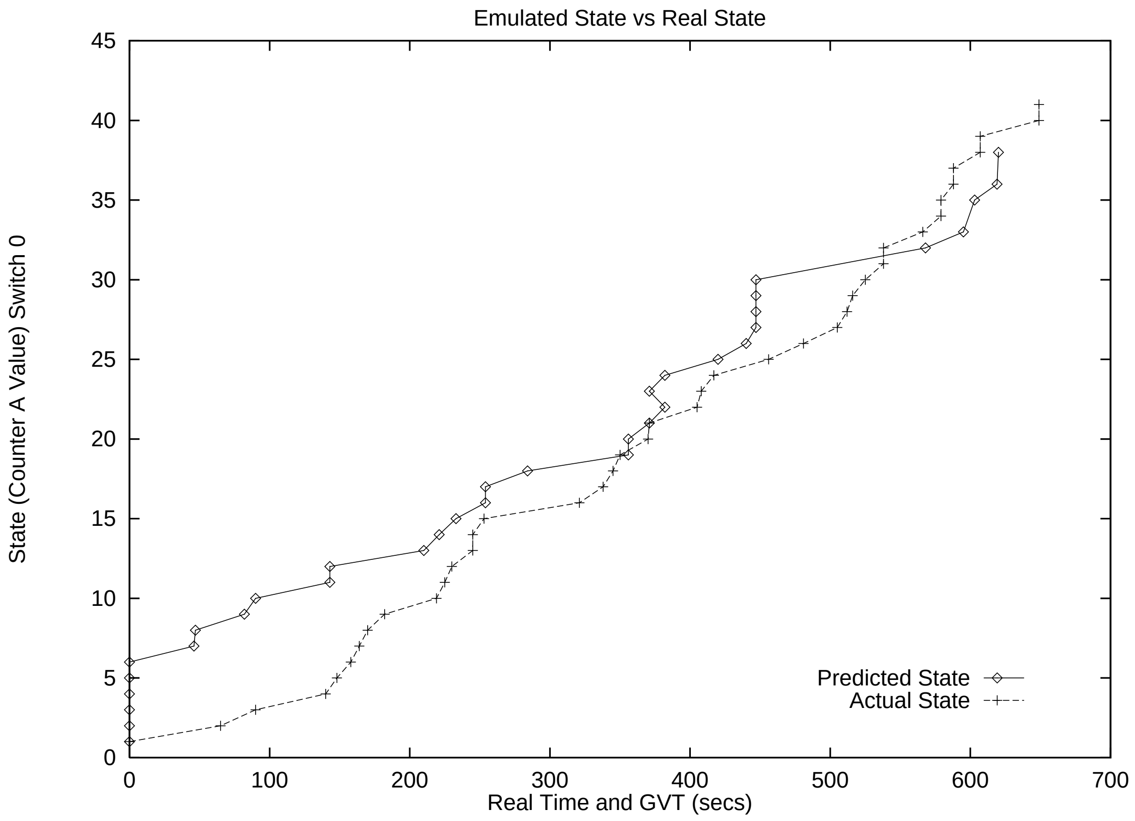


Figure 5.20: State (5, 10, 1)

failure rollbacks as shown in Figure 5.21. This is expected since the tolerance has been reduced from 10 to 3. The cluster of causality rollbacks near the state verification rollbacks was expected. These clusters of causality rollbacks do not appear to significantly reduce the feasibility of the system. The real-time versus GVT plot as shown in Figure 5.21 shows much larger jumps as the Logical Processs were held back due to rollbacks. The entities had a larger variance in their hold times than the $(5, 10, 5)$ run. The state graph for this run is shown if Figure 5.22.

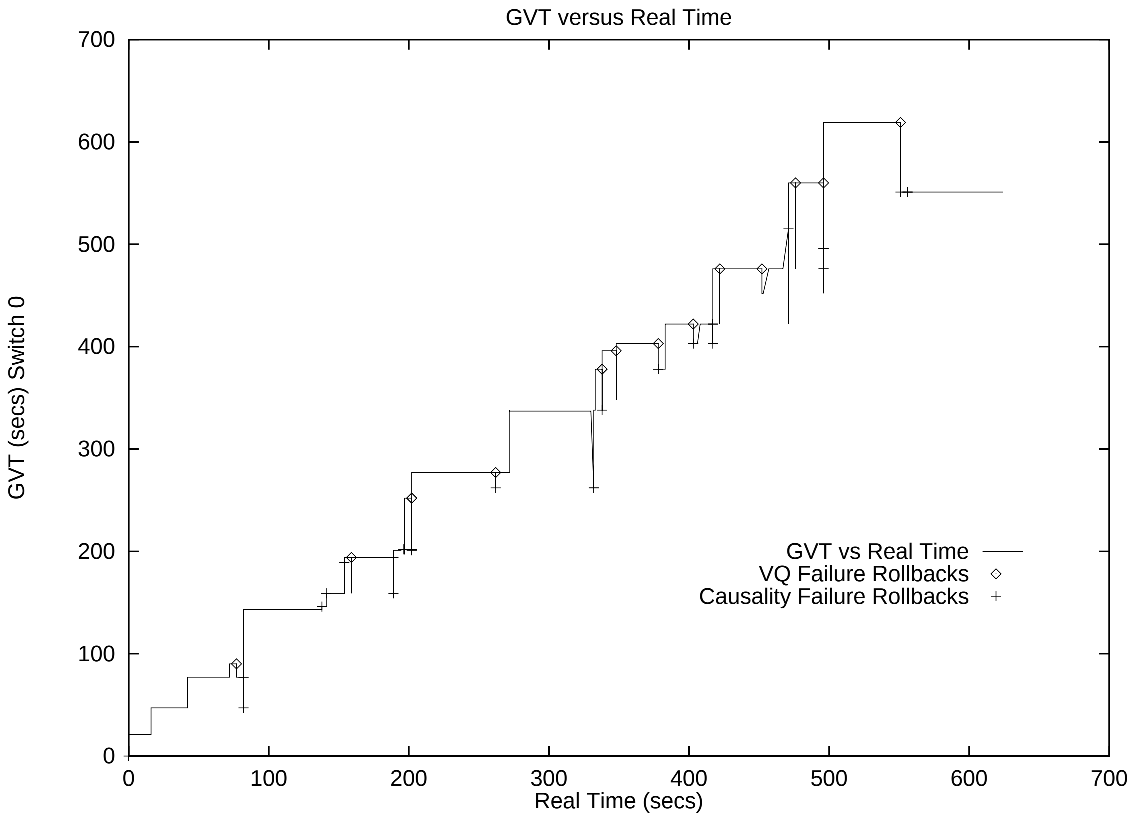


Figure 5.21: Rollbacks Due to State Verification Failure (5, 3, 5)

A $(400, 5, 5)$ run showed the GVT jump quickly to 400 and then gradually increase as the Sliding Lookahead Window maintained a 400 time unit lead as shown in Figure 5.23. The Logical Process hold times were shorter here than an any previous run. The state graph for this run is shown in Figure 5.24.

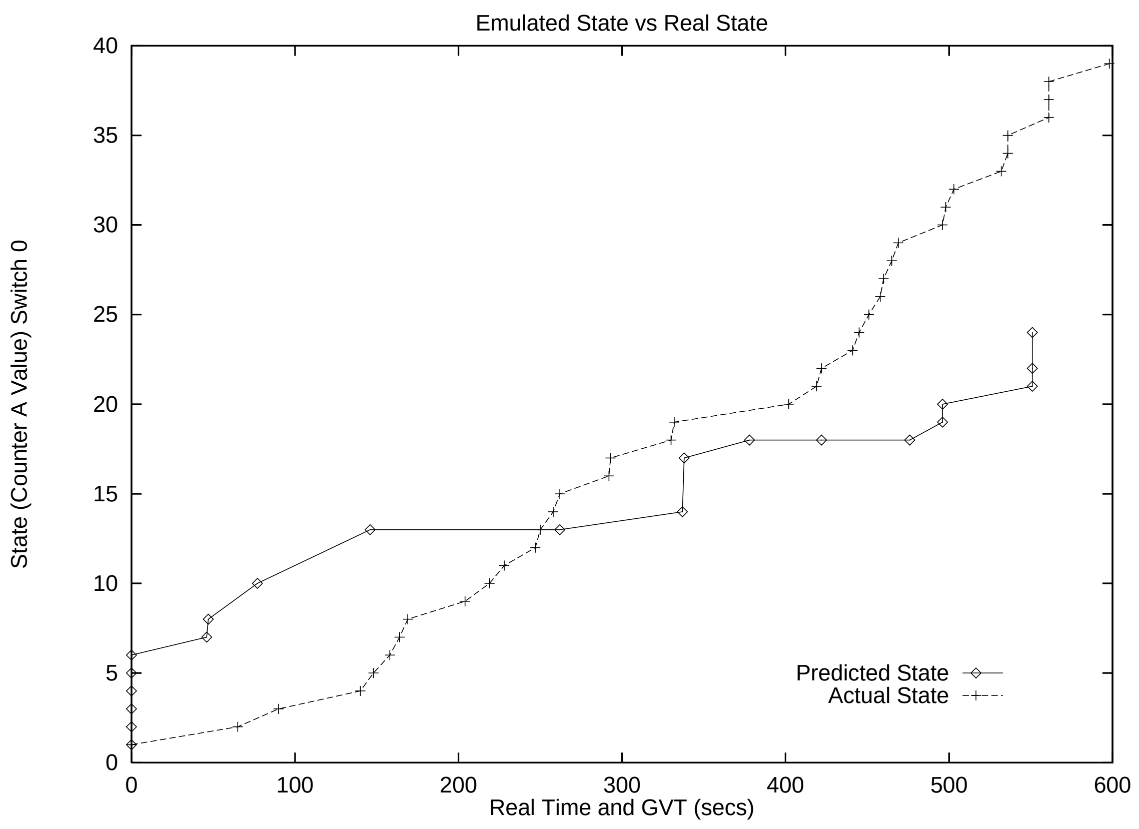


Figure 5.22: State (5, 3, 5)

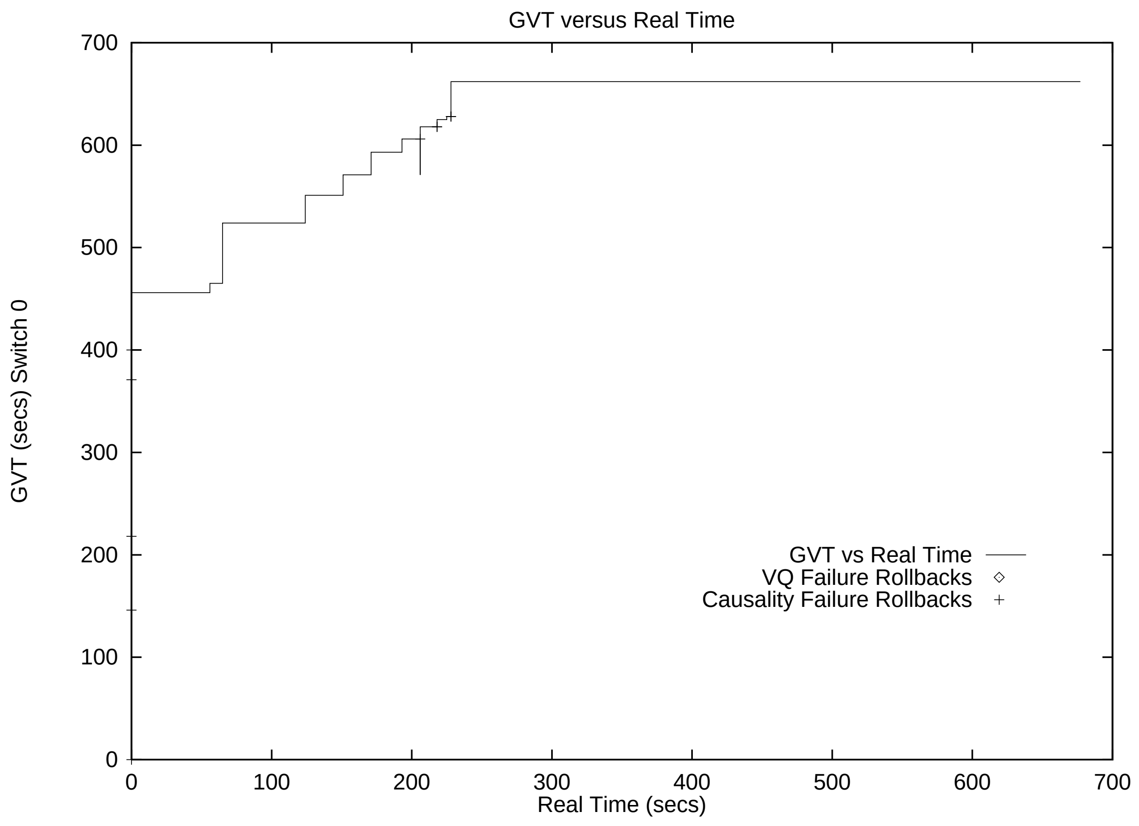


Figure 5.23: Rollbacks Due to State Verification Failure (400, 5, 5)

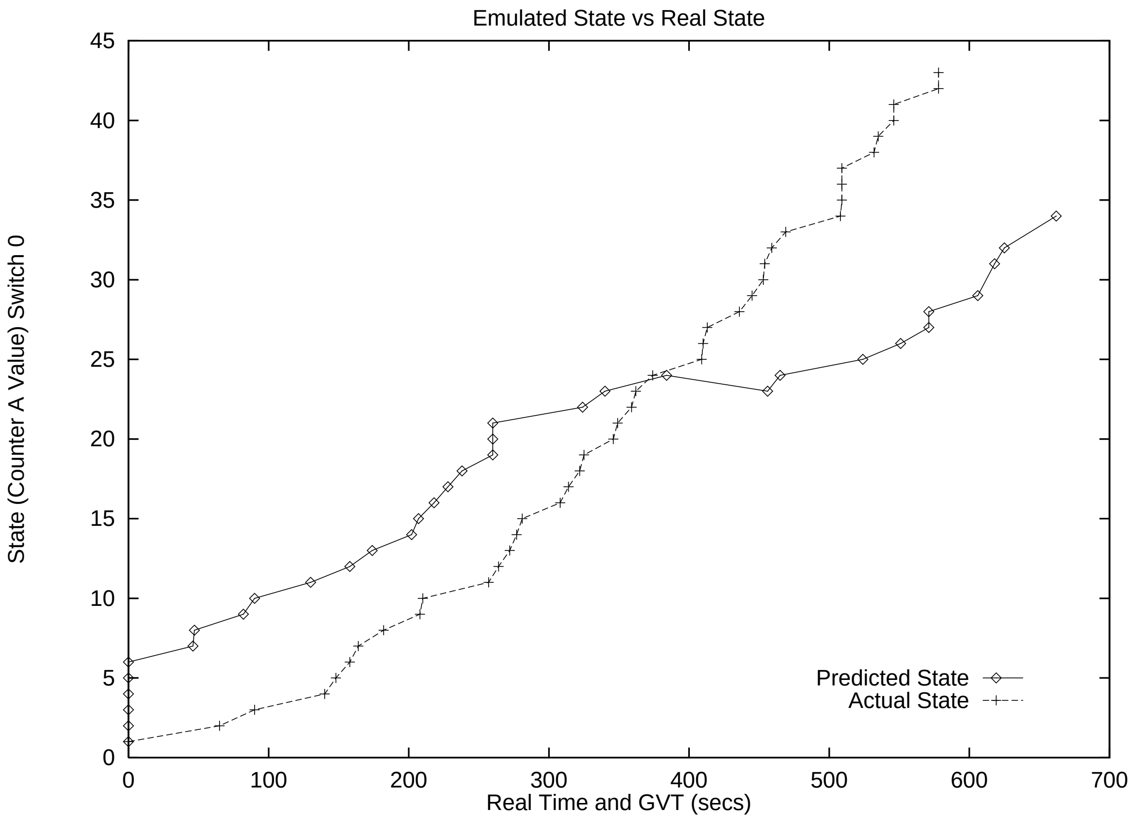


Figure 5.24: State (400, 5, 5)

5.8.3 Discussion of Predictive Management Simulation Results

Clearly, these results show the system to be stable with the introduction of state verification rollbacks. The overhead introduced by these rollbacks did not greatly impact the performance, because as previously shown in the GVT versus time graphs, Figures 5.16, 5.21 and 5.23, the system was always able to predict up to its lookahead time very quickly. An unstable system is one in which there exists exists enough rollbacks to cause the system to take longer than real-time to reach the end of the Sliding Lookahead Window.

The inter-Logical Process bandwidth overhead is estimated from the State Figures 5.17, 5.20, 5.22, and 5.24. The Predicted State value has been chosen to serve a dual purpose. It shows the virtual message load as well as serving as the predicted performance metric. Thus, the Predicted State value is also the number of virtual messages received by the switch. Since the rate of change of the state counter is the rate of packets entering a switch, the slope of the Predicted State curve in each state graph shows the amount of virtual message overhead. This does not include anti-messages since anti-messages are used only to cancel the effects of non-causal messages and are not counted. All the simulation runs had less than 4 anti-messages, or less than 6% more than the virtual message overhead, except for the (5,3,5) simulation. The (5,3,5) simulation generated a total of 17 anti-messages which is almost 28% more than the virtual message overhead. This is expected given the narrow tolerance for the (5,3,5) run.

As previously mentioned, the overhead metric of interest for Virtual Network Configuration in a predictive management environment is the amount of polling bandwidth, since polling status values from managed devices uses the same bandwidth that is being managed. In most standards based approaches, network management stations are sampling counters in managed entities which simply increment in value until they roll over. A management station which is simply plotting data will have some fixed polling interval and record the absolute value of the difference in value of the counter. Such a graph is not a perfectly accurate representation of the data, it is merely a statement that sometime within a polling interval the counter has monotonically increased by some amount. Spikes in this data, which may be very important to the current state of the system, may not be noticed if the polling interval is too long such that a spike followed by low data values expected out to a normal or low value. Our goal is to determine the minimum polling interval required to accurately represent the data.

From the information provided by the predictive management system, a polling interval which provides the desired degree of accuracy can be determined and dynamically adjusted; however, the cost must be determined. An upper limit on the number of systems which can be polled is $N \leqslant \frac{T}{\Delta}$ where N is the number of devices capable of being polled, T is the polling interval, and Δ is the time required for a single poll. Thus although the data accuracy will be constrained by this upper limit, taking advantage of characteristics of the data to be monitored can help distribute the polling intervals efficiently within this constraint. Assume that Δ is a calculated and fixed value, as is N. Thus this is a lower bound on the value of $T \geqslant \Delta N$.

The overhead bandwidth required for use by the management system to perform polling is shown in Equation 5.3. The packet size will vary depending upon whether it is an Simple Network Management Protocol or Common Management Information Protocol packet and the Management Information Base object(s) being polled. The number of packets varies with the amount of management data requested. Let P be the number of packets, S be the bits/packet, N be the number of devices polled, and T be the polling period. Bw is the total available bandwidth and Bw_{oh} is the overhead bandwidth of the management traffic.

It may be desirable to limit the bandwidth used for polling system management data

$$Bw_{oh}\% = \frac{100.0PNSBw}{T} \tag{5.3}$$

to be no more than a certain percentage of total bandwidth. Thus the optimum polling interval will use the least amount of bandwidth while also maintaining the least amount of variance due to error in the data signal. All the required information to maintain the cost versus accuracy at a desired level is provided by the predictive network management system.

5.9 Virtual Network Configuration Experimental Validation Summary

This chapter has presented the results of a Virtual Network Configuration enhanced Rapidly Deployable Radio Network orderwire. The experimental validation of Virtual Network Configuration (VNC) examined simulations and implementations of the Virtual Network Configuration algorithm in order to determine the speedup (η) and bandwidth overhead (β) of the Virtual Network Configuration algorithm. The experimental validation also satisfied an existence proof which demonstrates the feasibility Virtual Network Configuration.

The implementation of the driving process is critical for Virtual Network Configuration, therefore this chapter began with a discussion of an ideal implementation for the driving process for Rapidly Deployable Radio Network position prediction. This indicated the accuracy that could be expected from a position prediction process. However, the implementation of the driving process for the experimental validation in this chapter simulated Global Positioning System (GPS) input. By simulating the Global

Positioning System, the amount of error in virtual messages was accurately controlled.

Virtual Network Configuration (VNC) was experimental validated in a mobile network configuration environment and in a simulated predictive network management environment. In the mobile network environment, a portion of the Rapidly Deployable Radio Network (RDRN) Network Control Protocol (NCP) had been enhanced with Virtual Network Configuration. Specifically, the task which processes **USER_POS** messages pre-computes and caches beam tables. The measurements of the Virtual Network Configuration enhanced Rapidly Deployable Radio Network (RDRN) Network Control Protocol (NCP) and a discussion of the results is in Section 5.3. In the simulated predictive network management environment, a standards based management system enhanced with Virtual Network Configuration was simulated with Maisie [6].

The results of the experimental validation in this section have shown that the Virtual Network Configuration algorithm can be stable. In other words, rollback cause by both out-of-tolerance and out-of-order messages do not cause the system predict events in the past. The results in the section also indicated that bandwidth overhead was not significantly more than twice the non-Virtual Network Configuration bandwidth. Also, this validation has shown that a system enhanced with Virtual Network Configuration can be used for fault prediction rather than to obtain speedup as in the Virtual Network Configuration Network Control Protocol (NCP) experimental implementation.

Chapter 6

Conclusion

6.1 Attributes of Virtual Network Configuration Performance

The focus of this research has been the extension of optimistic distributed simulation techniques to speedup the configuration of the wireless Asynchronous Transfer Mode network developed by the Rapidly Deployable Radio Network project. This research has resulted in a new algorithm for predictive real time systems which utilizes results from optimistic distributed simulation and mobile network protocols. The analysis and results have shown this to be a feasible method for performance enhancement in a wireless environment where prediction of future location is used to speedup configuration in the physical and higher protocol layers. The goal of Virtual Network Configuration is to provide accurate predictions quickly enough so that the results are available before they are needed. Without taking advantage of parallelism, a less sophisticated algorithm than Virtual Network Configuration could run ahead of real-time and cache results for future use. However, simply predicting and caching results ahead of time does not fully utilize inherent parallelism as long as messages between Logical Processs remain strictly synchronized. Strict synchronization means that processes must wait until all messages are insured to be processed in order. This research has shown how the cost of rollback overhead relates to the speedup gained using the Virtual Network Configuration algorithm.

Examples of physical and protocol layers which benefit from Virtual Network Configuration have been presented. These include beamforming, Asynchronous Transfer Mode topology calculation, Asynchronous Transfer Mode Address Resolution Protocol service, Next Hop Resolution Protocol, IP routing and Private Network-Network Interface. A pro-active network management application has also been explored. It should be noted that the Virtual Network Configuration algorithm has applications in any real-time system where information about the future state of the system can improve performance.

The analysis of Virtual Network Configuration has focused on three measurements: speedup, bandwidth overhead, and accuracy. These are a function of out-of-order message rate, out-of-tolerance state value rate, rate of virtual messages entering the system, task execution time, task partitioning into Logical Processs, rollback overhead, prediction accuracy as a function of distance into the future which predictions are attempted, and the effect of parallelism and optimistic synchronization. All of these factors have been analyzed. It has been shown both analytically and through experimental validation that significant speedup can be gained with Virtual Network Configuration while the worst case bandwidth overhead can be kept to slightly more than two times the bandwidth without Virtual Network Configuration and even less with the real message optimization.

The Virtual Network Configuration algorithm developed in this research will become increasingly important. As the physical layer for wireless technology improves, bandwidth will become cheaper. While more bandwidth becomes available, the trend is for smaller geographical cell sizes for wireless systems, thus increasing the probability and overhead associated with handoff. As shown in this research, as the utility for bandwidth goes down relative to the utility for speedup, the overall utility of Virtual Network Configuration increases.

6.2 Future Work

The cached information in each Logical Process contains useful information about the accuracy of the driving process. The difference between the actual state and the cached state can be fed back to the driving process in order to correct the prediction algorithm. An interesting technique to accomplish this could utilize Perturbation Analysis [43]. The technique of Perturbation Analysis allows a great deal more information to be obtained from a simulation execution than explicitly collected statistics. It is particularly useful for finding the sensitivity information of simulation parameters from the sample path of a single simulation run. In a practical sense, sensitivity analysis of the Virtual Network Configuration message or state values qualified with a tolerance would be useful information in the system design. However, this author has not found any work applying Perturbation Analysis to optimistic simulation methods in a fundamental manner. Knowledge of the sensitivity to error at each Logical Process gained through Perturbation Analysis could be used to examine the accumulation of error throughout the Virtual Network Configuration system. The tolerances could be automatically adjusted based on this information similar to back-propagation in a neural network. The system could be enhanced with a simultaneous learn phase of operation where the cause of a rollback is identified and corrected automatically by weighting input message values or providing feedback to the driving process(es).

It would also be interesting to study the effect of self-similarity and emergence in the virtual message values generated by the driving process on the operation of the Virtual Network Configuration algorithm and on Time Warp algorithms in general. This research assumes that the driving process will predict future results with greater accuracy as real time approaches the time of the predicted event. However, the predictions which appear in the state queue may be chaotic or self-similar over time. This would certainly be the case if Virtual Network Configuration is applied towards predicting LAN or Asynchronous Transfer Mode switch traffic, both of which are known to be self-similar.

Active networks, [2, 111] are communications networks which view their contents not simply as packets to be transported, but also as programs which may be interpreted by intermediate nodes within the network [64, 11]. Thus, active networks become an enabling mechanism for dynamically loading Virtual Network Configuration predictive processes into all network nodes including fixed and wireless networks. In the future, active data which implements the predictive algorithm in Virtual Network Configuration may be supplied by network vendors as commonly as an Simple Network Management Protocol Management Information Base is supplied today. If we view the Virtual Network Configuration algorithm in an active network environment, the Virtual Network Configuration messages may carry programs which are interpreted by intermediate nodes rather than passive messages. The algorithm then becomes much more flexible and powerful. Logical processes may be created or destroyed as the algorithm executes and the effect of a rollback becomes much more complex and interesting. The ideas introduced in this research, particularly self-predicting and emergent systems, are clearly a rich area for more research.

Appendix A

VNC Implementation

A.1 Overview of the VNC API

This section discusses enhancing a Physical Process (PP) with Virtual Network Configuration. The process descriptions below use the notation for Communicating Sequential Processes (CSP) [44]. In CSP “X?Y” indicates process X will wait until a valid message is received into Y, “X!Y” indicates X sends message Y. A guard statement is represented by “$X \rightarrow Y$” which indicates that condition X must be satisfied in order for Y to be executed. Concurrent operation is indicated by “X||Y” which means that X operates in parallel with Y. A “*” preceding a statement indicates that the statement is repeated indefinitely. An alternative command, represented by “X□Y”, indicates that either X or Y may be executed assuming any guards (conditions) that they may have are satisfied. If X and Y can both be executed, then only one is randomly chosen to execute. A familiar example used to illustrate CSP is shown in Algorithm 3. This is the bounded buffer problem in which a finite size buffer requests more items from a consumer only when the buffer will not run out of capacity.

Assume a working Physical Process abstracted in Algorithm 4 where *S* and *D* represent the source and destination of real and virtual messages. Algorithm 5 shows the Physical Process converted to a Virtual Network Configuration Logical Process operating with a monotonically increasing Local Virtual Time. Note that the actual Virtual Network Configuration API function names are used, however, all the function arguments are not shown in order to simplify the explanation. Each function is described in more detail later. First the Virtual Network Configuration API is initialized by vncinit() as shown in Algorithm 5 Line 2. Then input messages are queued in the Receive Queue by recvm(). In non-rollback operation (Lines 6 to 11) the function getnextvm() returns the next valid message from the Receive Queue to be processed by the Physical Process. When the Physical Process has a message to be sent, the message is place in the Send Queue by sendvm(). While messages are flowing through the process, the process saves its state periodically. Normal operation of the Virtual Network Configuration as just described may be interrupted by a rollback. If recvm() returns a non-zero value (Line 4), then either an out-of-order or out-of-tolerance message has been received. In order to perform the rollback, getstate() is called to return the proper state to which the process must rollback. It is the applications responsibility to insure that the data returned from getstate() properly restores the process state. Anti-messages are sent by repeatedly calling rbq() (Line 5) until rbq() returns a null value. With each call of rbq(), an anti-message is returned which is sent to the destination of the original message.

Algorithm 3: Classical Bounded Buffer in CSP.

```
(1)   X::
(2)   buffer:(0..9) portion;
(3)   in,out:integer; in := 0; out := 0;
(4)    *[in < out + 10; producer?buffer(in mod 10) →
(5)      in := in + 1;
(6)    □ out < in; consumer?more() →
(7)      consumer!buffer(out mod 10); out := out + 1; ]
```

Algorithm 4: A Physical Process before VNC Added.

```
(1)     PP::
(2)     *[S?input;
(3)       output := process(input);
(4)        D!output]
```

Algorithm 5: A Physical Process with VNC Added.

```
(1)     PP::
(2)     initvnc();
(3)     *[S?input;
(4)      [recvm(input)!=0 → getstate();
(5)      *[rbq()!=NULL → D!rbq()] □
(6)       [recvm(input)==0 →
(7)        savestate();
(8)        input := getnextvm();
(9)        output := process(input);
(10)       sendvm(output);
(11)       D!output]
(12)     ]
(13)    ]
```

A.2 VNC API Implementation

Figure A.1 lists the files and their contents for the Virtual Network Configuration API. The following functions can be added to a Physical Process in order to implement Virtual Network Configuration.

A.2.1 External Library Declarations

```
extern struct vm *vminq; /* receive queue head */
extern struct vm *vmoutq; /* output queue head */
extern struct sq *stateq; /* state queue head */
```

vc.h	Virtual Network Configuration include file.
vcrec.c	Receive a message, determine whether virtual or real, rollback
vcsnd.c	Send a virtual message
vcsendq.c	All queue related functions
vcroll.c	Roll back to given time
vcstate.c	Maintain virtual state
vctime.c	Local virtual time maintenance functions
libvnc.a	Library of these functions

Figure A.1: VNC API Files.

A.2.2 void initvc (struct vcid *id, char *name, int num)

This function is required for initializing Virtual Network Configuration. The "id" pointer is used for all other Virtual Network Configuration functions in order to identify the Logical Process. The "name" and "num" are input to uniquely identify the Logical Process. Multiple Logical Processs may be run with the same Unix process.

A.2.3 time_t getlvt (struct vcid *id)

This function Returns the current Local Virtual Time of Logical Process "id".

A.2.4 int setlookahead(struct vcid *id, time_t t)

This function sets the maximum amount of time into the future (Λ) that the Local Virtual Time for this process can go.

A.2.5 long recvm (struct vcid *id, int anti, long rec, long snd, char *src, char *dst, int ptype, char *m, int msize, int (*comp_pred) (), int fd, int sg)

This function should be called each time a message arrives. The function comp_pred() is user defined and described in detail in the next section. The function recvm() handles both real and virtual messages, and will detect if rollback is necessary. It returns time_t if temporal order of messages is violated or a real message is out of tolerance, otherwise it returns 0. If non-zero, the return value is the time to which the process must rollback. The input arguments *anti*, *rec*, *snd*, *src*, *dst* are the message values after a message has been depacketize. The input variable *msize* is the size of the packetized packet and *m* is a pointer to the structure placed on the Receive Queue. The input variables *fd* and *sg* are the Global Positioning System file descriptor and whether the Global Positioning System receiver is simulated or real.

A.2.6 int comp_pred (struct vcid *vncid, struct vm *s1, struct vm *s2, struct sq *stq)

This function is passed into recvm() and is user defined. It should return whether the State Queue differs beyond a tolerable amount (Θ) from the predicted value of the State Queue. If the actual and predicted State Queue values differ beyond Θ, comp_pred() should return true. The input *stq* is a pointer to the State Queue. The input arguments *s1* and *s2* are the current real and predicted virtual messages respectively. These are currently not used to determine rollback, however, *s1* and *s2* may be useful in future modifications to the algorithm.

A.2.7 struct vm *getnextm (struct vcid *id)

This function is called to determine the next message to process. This should be the only way the physical process receives input messages. This insures that all messages are processed first by the Virtual Network Configuration library functions. Note that the message space is freed after this call so the value must be saved immediately into a local variable.

A.2.8 int sendvm (struct vcid *id, int anti, long rec, long snd, char *src, char *dst, int ptype, char *m, int msize)

This function should be called whenever messages are sent from the process. It stores the message in the Send Queue and records the Local Virtual Time at the time the message was sent for rollback purposes.

A.2.9 struct vm *rbq (struct vcid *id, struct vm *vncpkt, long t)

This function should be called repeatedly when recvm() returns a non-zero value indicating a rollback. The *vncpkt* output pointer is an anti-message. rbq() should be called until it returns null. Because the messages returned are anti-messages, there is no need to store them in the Send Queue, thus these message should be sent without using sendvm().

A.2.10 struct sq *savestate (struct vcid *id, void *m, int msize)

This function should be called periodically to save the Physical Process state for rollbacks. The input variable *m* holds the state which is dynamically allocated based on the input *msize*. Note that the more often state is saved, the less far back rollback has to occur. However, this is a tradeoff between storage and efficiency.

A.2.11 struct sq *getstate (struct vcid *id, long snd)

This function will return the state which was saved closest to the input variable *snd*. This function is used after a rollback to restore a valid state. The time is obtained from the return value of recvm().

A.3 The VNC Message Receive Function

There are two small but important sections of Network Control Protocol code which are illustrate the use of the Virtual Network Configuration library developed by the author of this study. The reason for including the actual code in this appendix is to demonstrate the simplicity and ease of enhancing a system with Virtual Network Configuration. The first section of code receives an Network Control Protocol packet and processes the packet in Virtual Network Configuration enhanced mode. At line 28 the packet is received in non-blocking mode from the orderwire packet radio. The Virtual Network Configuration related information is stripped form the packet at line 33. This information is the send-time, receive-time, and anti-message indicator. The next line handles the bulk of the Virtual Network Configuration processing. The function recvm() checks for causal and out-of-tolerance rollback, updates the local virtual time, manages the receive queue, and checks for message annihilation. Lines 37 through 46 send anti-messages because a rollback has occurred.

```
#include <stdio.h>
#include <ctype.h>
#include "switch.h"
```

```
#include "shared.h"
#include "vnc.h"

extern struct shared_data *sharedm;

static char *rcs = "$Id:  vncgetpkt.tex,v 1.2 1997/05/06 16:05:06 sbush Exp sbush $";

/*
 * Implement the VNC algorithm for incoming packets.
 * struct sq *st is a union of all states (see vnc.h).
 */

int
vncgetpkt (struct vcid *vncid, struct sq *st, int (comp_pred)
     (struct vcid * vncid, struct vm * s1, struct vm * s2, struct sq * stq),
           int *gfd, int *efd, int *sfd, struct owPacket *op)
{
  struct vm vncpkt;
  time_t rt;
  struct vm *nxtmsg;
  struct gvtS GVT;
  int pkt_typ;
  struct gps pos;                 /* Position packet and table */

  pkt_typ = getpkt (efd, sfd, op);        /* read the next message */
  if (sharedm->UseVNC && pkt_typ != NOPACKET)
    {
      /* display time and info */
      displaynow (vncid, "VNC Packet Received");
      vtpacketize (op->m, &vncpkt.anti, &vncpkt.snd_tm, &vncpkt.rec_tm);
      rt = recvm (vncid, vncpkt.anti, vncpkt.rec_tm,         /* check for rollback */
                vncpkt.snd_tm, op->m, strlen (op->m), (int *) comp_pred,
                *gfd, (int) 1);
      if (rt)
        {
          displaynow (vncid, "Rollback");        /* display time and info */
          bcopy (getstate (vncid, rt), st, sizeof (struct sq));
          while ((nxtmsg = (struct vm *) getnextm (vncid)) == NULL)
            {
              displaynow (vncid, "Sending Anti-Message");    /* send anti-messages */
              OrderWireTransmit (efd, sfd, op->ptype, op->dst, op->m, PPP, op->src);
            }
        }
      nxtmsg = getnextm (vncid);              /* return next message by order of
                                               * received time.
                                               */
      parsepacket (nxtmsg->m, op);            /* parse the message */
      pkt_typ = op->ptype;

      printf ("(vncgetpkt) before deletevminq\n");
      fflush (stdout);
      fflush (stderr);
      deletevminq (vncid, nxtmsg);
    }
  printf ("(vncgetpkt) returning %s\n", displayPktTyp (pkt_typ));
  fflush (stdout);
  fflush (stderr);
  return pkt_typ;
}
```

A.4 The VNC Message Send Function

The Virtual Network Configuration enhanced Network Control Protocol transmit function is shown below. This function simply sets the Virtual Network Configuration information appropriately and stores the packet in the Send Queue (QS) by calling sendvc().

```
#include <stdio.h>
#include <sys/types.h>
#include "switch.h"
#include "shared.h"
#include "vnc.h"

extern struct shared_data *sharedm;

static char *rcs = "$Id:  vnctransmit.tex,v 1.2 1997/05/06 16:05:06 sbush Exp sbush $";

/*
 * Implement the VNC algorithm for outgoing orderwire packets.
 */

int
vncOrderWireTransmit (struct vcid *vncid, int gfd, int efd, int sfd, int ptype,
                      char *dest, char *m, int p, char *callsign)
{
  int res;
  struct vm vncpkt;
  time_t rt;
  struct vm *nxtmsg;
  struct gvtS GVT;
  struct gps pos;              /* Position packet and table */
  if (sharedm->UseVNC)
    {
      vncpkt.anti = 0;
      vncpkt.snd_tm = getlvt (vncid);
      vncpkt.rec_tm = getgpstime (gfd, 1);
      vtdepacketize (m, vncpkt.anti, vncpkt.snd_tm, vncpkt.rec_tm);
      sendvm (vncid, vncpkt.anti, vncpkt.rec_tm, vncpkt.snd_tm, m, strlen (m));
    }
  return OrderWireTransmit (efd, sfd, ptype, dest, m, p, callsign);
}
```

Appendix B

NCP and VNC MIBS

B.1 The RDRN Simple Network Management MIB

The following Simple Network Management Protocol (SNMP) Management Information Base (MIB) is used by the RDRN Network Control Protocol (NCP) and Link Management Daemon (LMD).

```
RDRN−NCP−MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE−IDENTITY, OBJECT−TYPE, experimental,
    Counter32, TimeTicks
        FROM SNMPv2−SMI
    DisplayString
        FROM SNMPv2−TC;

rdrnNcpMIB MODULE−IDENTITY
     LAST−UPDATED "9701010000Z"
     ORGANIZATION "KU TISL"
     CONTACT−INFO
              "Steve Bush sbushtisl.ukans.edu"
     DESCRIPTION
             "Experimental MIB modules for the Rapidly Deployable
              Radio Network (RDRN) Project Network Control
              Protocol (NCP)."
     ::= { experimental ncp(75) 2 }
```

```
--
-- NCP Initialization Group
--

initialization OBJECT IDENTIFIER ::= { rdrnNcpMIB 1 }

initContact OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "NCP Agent Developer, but hopefully NOT maintainer."
        ::= { initialization 1 }

initEntity OBJECT-TYPE
        SYNTAX INTEGER {
                edgeSwitch(0),
                remoteNode(1)
        }
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "Describes the RDRN node component."
        ::= { initialization 2 }

initDebug OBJECT-TYPE
        SYNTAX INTEGER (0..100)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "Logging level. Zero is off; the higher the number, the
                 more logging is enabled."
        ::= { initialization 3 }

initEncrypt OBJECT-TYPE
        SYNTAX INTEGER {
                clear(0),
                des(1)
        }
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "Indicates whether data encryption used. Both endpoints of
                 a packet radio link MUST have this object set identically
                 for proper communication."
        ::= { initialization 4 }

initVNC OBJECT-TYPE
        SYNTAX INTEGER {
                vncEnabled(1),
                sequentional(0)
        }
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates whether this system is running using
                 Virtual Network Configuration enhanced mode,
                 or sequential operation."
        ::= { initialization 5 }

initStacks OBJECT−TYPE
        SYNTAX INTEGER {
                noStacks(1),
                useStacks(0)
        }
        MAX−ACCESS read−write
        STATUS current
        DESCRIPTION
                "Indicates whether link level stacks will be created.
                 Primarily used for testing purposes."
        ::= { initialization 6 }

initRadioPort OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates the packet radio tty port."
        ::= { initialization 7 }

initGpsPort OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates the GPS tty port."
        ::= { initialization 8 }

initSpeed OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates the assumed initial speed. Primarily used for
                 simulated GPS."
        ::= { initialization 9 }

initDirection OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates the assumed initial direction. Primarily used for
                 simulated GPS."
        ::= { initialization 10 }

initXPos OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates the assumed initial X axis position. Primarily
                 used for simulated GPS."
        ::= { initialization 11 }

initYPos OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates the assumed initial Y axis position. Primarily
                 used for simulated GPS."
        ::= { initialization 12 }

initSimGps OBJECT−TYPE
        SYNTAX INTEGER {
                gpsReceiver(0),
                gpsSimulator(1)
        }
        MAX−ACCESS read−write
        STATUS current
        DESCRIPTION
                "Indicates whether the GPS receiver is being read or
                 the GPS simulation code is being used. This value
                 can be set while running without harm. Changing
                 from gpsReceiver to gpsSimulator will cause a smooth
                 transition based on last known direction and speed.
                 Transitioning from gps Simulator to gpsReceiver may
                 cause a large jump in position."
        ::= { initialization 13 }

initTcpEmulation OBJECT−TYPE
        SYNTAX INTEGER {
                packetRadio(2),
                tcpEmulation(1),
                packetRadioAndTcpEmulation(3)
        }
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates whether packet radios or TCP−IP simulation of
                 packet radios is currently in use. This value cannot
                be set because the tcp emulation requires knowledge
                of ip addresses of all hosts in the simulated RDRN
                system and all though this could be set on the fly
                through a MIB table, I don't feel it's worth the
                effort to develop at this point."
        ::= { initialization 14 }

--
-- NCP Configuration Group
--

configuration OBJECT IDENTIFIER ::= { rdrnNcpMIB 2 }

configTimeout OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
               "Indicates the timeout during edge switch reconfiguration.
                Should be set long enough for all edge switches to
                respond."
        ::= { configuration 1 }

configMaxVel OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
               "Indicates the maximum of a uniformly distributed velocity
                for simulated GPS operation."
        ::= { configuration 2 }

configSpeed OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
               "Indicates the initial speed for simulated GPS operation.
                Same as initSpeed for historical reasons, e.g. this one
                set through a file while the other is set via arguments."
        ::= { configuration 3 }

configDirection OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
               "Indicates the initial direction in degrees for simulated
                GPS operation. Same as initDirection for historical
                reasons. This one is read from a file while the other
                is set as a program argument during startup."
        ::= { configuration 4 }

configUseRealTopology OBJECT−TYPE
        SYNTAX INTEGER (0..1)
        MAX−ACCESS read−write
        STATUS current
        DESCRIPTION
                "Indicates whether edge switch topology code is called.
                 Currently this code uses MatLab and can be quite slow."
        ::= { configuration 5 }

configCallsign OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates this nodes packet radio callsign."
        ::= { configuration 6 }

configKey OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Indicates the current DES key. Probably not secure to
                 put this here, but useful for debugging."
        ::= { configuration 7 }

configUserSwitch OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "If this is a remote node, this is the associated edge
                 switch."
        ::= { configuration 8 }

configUpdatePeriod OBJECT−TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX−ACCESS read−write
        STATUS current
        DESCRIPTION
                "Indicates the remote node's user position update rate."
        ::= { configuration 9 }

configDistanceTolerance OBJECT−TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX−ACCESS read−write
        STATUS current
        DESCRIPTION
                "Indicates the amount of movement allowed before
                beam angles must be recomputed."
        ::= { configuration 10 }

configAntennaAngle OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "Indicates the direction of the physical position
                 of the linear array from pointing North."
        ::= { configuration 11 }

--
-- Current NCP AX.25 Connection Group
--

radioStreams OBJECT IDENTIFIER ::= { rdrnNcpMIB 3 }

radioStreamsTable OBJECT-TYPE
        SYNTAX SEQUENCE OF RadioStreamsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "Table of packet currently established packet radio streams
                 and callsigns."
        ::= { radioStreams 1 }

radioStreamsEntry OBJECT-TYPE
        SYNTAX RadioStreamsEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "Table of packet currently established packet radio streams
                 and callsigns."
        INDEX { radioStreamsID }
        ::= { radioStreamsTable 1 }

RadioStreamsEntry ::= SEQUENCE {
        radioStreamsID
                DisplayString,
        radioStreamsCallsign
                DisplayString
        }

radioStreamsID OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The Packet Radio stream identifier for this
                 ARQ AX.25 connection."
        ::= { radioStreamsEntry 1 }

radioStreamsCallsign OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "This is the callsign of the packet radio with
                 the associated stream identifier."
        ::= { radioStreamsEntry 2 }

−−
−− Known Edge Switch Group
−−

edgeSwitchNetwork OBJECT IDENTIFIER ::= { rdrnNcpMIB 4 }

edgeSwitchTable OBJECT−TYPE
        SYNTAX SEQUENCE OF EdgeSwitchEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "Table of RDRN edge switches known by this edge switch."
        ::= { edgeSwitchNetwork 1 }

edgeSwitchEntry OBJECT−TYPE
        SYNTAX EdgeSwitchEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "Table of RDRN edge switches known by this edge switch."
        INDEX { edgeSwitchCallsign }
        ::= { edgeSwitchTable 1 }

EdgeSwitchEntry ::= SEQUENCE {
        edgeSwitchCallsign
                DisplayString,
        edgeSwitchHostname
                DisplayString,
        edgeSwitchATMAddr
                DisplayString,
        edgeSwitchLatDeg
                DisplayString,
        edgeSwitchLatMin
                DisplayString,
        edgeSwitchLatDir
                DisplayString,
        edgeSwitchLonDeg
                DisplayString,
        edgeSwitchLonMin
                DisplayString,
        edgeSwitchLonDir
                DisplayString,
        edgeSwitchXPos
                DisplayString,
        edgeSwitchYPos
                DisplayString,
        edgeSwitchSpeed
                DisplayString,
        edgeSwitchDir
                DisplayString,
        edgeSwitchAvaliability
                INTEGER,
        edgeSwitchNSat
                INTEGER,
        edgeSwitchHDop
                INTEGER,
        edgeSwitchElevation
                DisplayString,
        edgeSwitchHeight
                DisplayString,
        edgeSwitchIndex
                DisplayString,
        edgeSwitchTime
                TimeTicks
}

edgeSwitchCallsign OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Edge Switch Callsign."
                ::= { edgeSwitchEntry 1 }

edgeSwitchHostname OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Edge Switch IP Hostname."
                ::= { edgeSwitchEntry 2 }

edgeSwitchATMAddr OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Edge Switch ATM Address."
                ::= { edgeSwitchEntry 3 }

edgeSwitchLatDeg OBJECT-TYPE
```

```
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch Latitude − degrees."
                ::= { edgeSwitchEntry 4 }

edgeSwitchLatMin OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch Latitude − minutes."
                ::= { edgeSwitchEntry 5 }

edgeSwitchLatDir OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch Latitude Hemisphere."
                ::= { edgeSwitchEntry 6 }

edgeSwitchLonDeg OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch Longitude − degrees."
                ::= { edgeSwitchEntry 7 }

edgeSwitchLonMin OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch Longitude − minutes."
                ::= { edgeSwitchEntry 8 }

edgeSwitchLonDir OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch Longitude − Hemisphere."
                ::= { edgeSwitchEntry 9 }

edgeSwitchXPos OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch Cartesian X Coordinate. Units are
                         meters."
                ::= { edgeSwitchEntry 10 }

edgeSwitchYPos OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Edge Switch Cartesian Y Coordinate. Units are
                         meters."
                ::= { edgeSwitchEntry 11 }

edgeSwitchSpeed OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Edge Switch current speed."
                ::= { edgeSwitchEntry 12 }

edgeSwitchDir OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Edge Switch current direction in degrees
                         0 degrees is North."
                ::= { edgeSwitchEntry 13 }

edgeSwitchAvaliability OBJECT-TYPE
                SYNTAX INTEGER (0..2)
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Edge Switch GPS quality information.
                         0 - fix not available
                         1 - Non differential GPS fix available
                         2 - Differential GPS fix available."
                ::= { edgeSwitchEntry 14 }

edgeSwitchNSat OBJECT-TYPE
                SYNTAX INTEGER (0..8)
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Edge Switch GPS number of satellites in use.
                         From 0 to 8 satellites."
                ::= { edgeSwitchEntry 15 }
```

```
edgeSwitchHDop OBJECT−TYPE
                SYNTAX INTEGER (0..999)
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch GPS horizontal dilution of position.
                         Varies from 1.0 to 99.9."
                ::= { edgeSwitchEntry 16 }

edgeSwitchElevation OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch current height with respect to sea
                         level in meters."
                ::= { edgeSwitchEntry 17 }

edgeSwitchHeight OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Edge Switch current geoidal height in meters."
                ::= { edgeSwitchEntry 18 }

edgeSwitchIndex OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Internal index for use in ES/RN associations."
                ::= { edgeSwitchEntry 19 }

edgeSwitchTime OBJECT−TYPE
                SYNTAX TimeTicks
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "GPS time at which this information was valid."
                ::= { edgeSwitchEntry 20 }

−−
−− Associated Remote Nodes Group
−−

remoteNodeNetwork OBJECT IDENTIFIER ::= { rdrnNcpMIB 5 }

remoteNodeTable OBJECT−TYPE
        SYNTAX SEQUENCE OF RemoteNodeEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "Table of RDRN remote nodes known by this edge switch."
        ::= { remoteNodeNetwork 1 }

remoteNodeEntry OBJECT−TYPE
        SYNTAX RemoteNodeEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "Table of RDRN remote nodes known by this edge switch."
        INDEX { remoteNodeCallsign }
        ::= { remoteNodeTable 1 }

RemoteNodeEntry ::= SEQUENCE {
        remoteNodeCallsign
                DisplayString,
        remoteNodeHostname
                DisplayString,
        remoteNodeATMAddr
                DisplayString,
        remoteNodeLatDeg
                DisplayString,
        remoteNodeLatMin
                DisplayString,
        remoteNodeLatDir
                DisplayString,
        remoteNodeLonDeg
                DisplayString,
        remoteNodeLonMin
                DisplayString,
        remoteNodeLonDir
                DisplayString,
        remoteNodeXPos
                DisplayString,
        remoteNodeYPos
                DisplayString,
        remoteNodeSpeed
                DisplayString,
        remoteNodeDir
                DisplayString,
        remoteNodeAvaliability
                INTEGER,
        remoteNodeNSat
                INTEGER,
        remoteNodeHDop
                INTEGER,
        remoteNodeElevation
                DisplayString,
        remoteNodeHeight
                DisplayString,
        remoteNodeIndex
                DisplayString,
        remoteNodeTime
                TimeTicks,
        remoteNodeDistance
                DisplayString
}

remoteNodeCallsign OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Callsign."
                ::= { remoteNodeEntry 1 }

remoteNodeHostname OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node IP hostname."
                ::= { remoteNodeEntry 2 }

remoteNodeATMAddr OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node ATM Address."
                ::= { remoteNodeEntry 3 }

remoteNodeLatDeg OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Latitude − degrees."
                ::= { remoteNodeEntry 4 }

remoteNodeLatMin OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Latitude − minutes."
                ::= { remoteNodeEntry 5 }

remoteNodeLatDir OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Latitude Hemisphere."
                ::= { remoteNodeEntry 6 }

remoteNodeLonDeg OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Longitude − degrees."
                ::= { remoteNodeEntry 7 }

remoteNodeLonMin OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Longitude − minutes."
                ::= { remoteNodeEntry 8 }

remoteNodeLonDir OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Longitude − Hemisphere."
                ::= { remoteNodeEntry 9 }

remoteNodeXPos OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Cartesian X Coordinate. Units are
                         meters."
                ::= { remoteNodeEntry 10 }

remoteNodeYPos OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node Cartesian Y Coordinate. Units are
                         meters."
                ::= { remoteNodeEntry 11 }

remoteNodeSpeed OBJECT−TYPE
                SYNTAX DisplayString
                MAX−ACCESS read−only
                STATUS current
                DESCRIPTION
                        "Remote Node current speed."
                ::= { remoteNodeEntry 12 }

remoteNodeDir OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Remote Node current direction in degrees
                         0 degrees is North."
                ::= { remoteNodeEntry 13 }

remoteNodeAvaliability OBJECT-TYPE
                SYNTAX INTEGER (0..2)
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Remote Node GPS quality information.
                         0 - fix not available
                         1 - Non differential GPS fix available
                         2 - Differential GPS fix available."
                ::= { remoteNodeEntry 14 }

remoteNodeNSat OBJECT-TYPE
                SYNTAX INTEGER (0..8)
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Remote Node GPS number of satellites in use.
                         From 0 to 8 satellites."
                ::= { remoteNodeEntry 15 }

remoteNodeHDop OBJECT-TYPE
                SYNTAX INTEGER (0..999)
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Remote Node GPS horizontal dilution of position.
                         Varies from 1.0 to 99.9."
                ::= { remoteNodeEntry 16 }

remoteNodeElevation OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Remote Node current height with respect to sea
                         level in meters."
                ::= { remoteNodeEntry 17 }
```

```
remoteNodeHeight OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Remote Node current geoidal height in meters."
                ::= { remoteNodeEntry 18 }

remoteNodeIndex OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Internal index for use in ES/RN associations."
                ::= { remoteNodeEntry 19 }

remoteNodeTime OBJECT-TYPE
                SYNTAX TimeTicks
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "GPS time at which this information was valid."
                ::= { remoteNodeEntry 20 }

remoteNodeDistance OBJECT-TYPE
                SYNTAX DisplayString
                MAX-ACCESS read-only
                STATUS current
                DESCRIPTION
                        "Distance of this remote node from its associated
                         base station. A value of 0.0 means that no base
                         station has been associated yet."
                ::= { remoteNodeEntry 21 }

--
-- GPS Specific Stats Group
--

gps OBJECT IDENTIFIER ::= { rdrnNcpMIB 6 }

gpsPosRead OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The total number of GPS position records that have
                 been read since initialization."
        ::= { gps 1 }

--
-- Packet Radio Specific Stats Group
```

```
--

packetRadio OBJECT IDENTIFIER ::= { rdrnNcpMIB 7 }

packetsIn OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The total number of NCP packets which have
                 been read since initialization."
        ::= { packetRadio 1 }

packetsOut OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The total number of NCP packets which have
                 been written to the packet radio since initialization."
        ::= { packetRadio 2 }

packetsBad OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The total number of NCP packets which have
                 been read and discarded as unreadable or incomplete
                 since initialization."
        ::= { packetRadio 3 }

--
-- NCP Level Specific Stats Group
--

ncpLevel OBJECT IDENTIFIER ::= { rdrnNcpMIB 8 }

ncpHandoffs OBJECT-TYPE
        SYNTAX Counter32
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The total number handoffs since since initialization."
        ::= { ncpLevel 1 }

--
-- Existing ATM Stacks Group
--

atmStacks OBJECT IDENTIFIER ::= { rdrnNcpMIB 9 }
```

```
atmStacksTable OBJECT−TYPE
        SYNTAX SEQUENCE OF AtmStacksEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "Table of currently active AHDLC stacks."
        ::= { atmStacks 1 }

atmStacksEntry OBJECT−TYPE
        SYNTAX AtmStacksEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "Table of currently active AHDLC stacks."
        INDEX { atmStacksATMaddr }
        ::= { atmStacksTable 1 }

AtmStacksEntry ::= SEQUENCE {
        atmStacksBeam
                INTEGER,
        atmStacksSlot
                INTEGER,
        atmStacksATMaddr
                DisplayString,
        atmStacksAhdlcState
                INTEGER
        }

atmStacksBeam OBJECT−TYPE
        SYNTAX INTEGER (0..8)
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "The beam number for this AHDLC stack."
        ::= { atmStacksEntry 1 }

atmStacksSlot OBJECT−TYPE
        SYNTAX INTEGER (0..256)
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "The slot number for this AHDLC stack."
        ::= { atmStacksEntry 2 }

atmStacksATMaddr OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "The ATM address of the current AHDLC stack."
```

```
          ::= { atmStacksEntry 3 }

atmStacksAhdlcState OBJECT−TYPE
          SYNTAX INTEGER {
                    down(0),
                    up(1)
          }
          MAX−ACCESS read−only
          STATUS current
          DESCRIPTION
                    "This is the state of the AHDLC layer."
          ::= { atmStacksEntry 4 }

−−
−− The Beam Coverage Group
−−

beamCoverage OBJECT IDENTIFIER ::= { rdrnNcpMIB 10 }

beamCovTable OBJECT−TYPE
          SYNTAX SEQUENCE OF BeamCovEntry
          MAX−ACCESS not−accessible
          STATUS current
          DESCRIPTION
                    "Table of beam coverage information."
          ::= { beamCoverage 1 }

beamCovEntry OBJECT−TYPE
          SYNTAX BeamCovEntry
          MAX−ACCESS not−accessible
          STATUS current
          DESCRIPTION
                    "Table of beam coverage information."
          INDEX { beamCovAngle }
          ::= { beamCovTable 1 }

BeamCovEntry ::= SEQUENCE {
          beamCovAngle
                    DisplayString,
          beamCovPower
                    DisplayString,
          beamCovSIR
                    DisplayString
          }

beamCovAngle OBJECT−TYPE
          SYNTAX DisplayString
          MAX−ACCESS read−only
          STATUS current
          DESCRIPTION
                    "The beam angle of this beam."
```

```
        ::= { beamCovEntry 1 }

beamCovPower OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "The power level of this beam."
        ::= { beamCovEntry 2 }

beamCovSIR OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "The signal to interference ratio for this beam."
        ::= { beamCovEntry 3 }

−−
−− The NCP Status Group
−−

status OBJECT IDENTIFIER ::= { rdrnNcpMIB 11 }

statState OBJECT−TYPE
        SYNTAX INTEGER {
                down(0),
                configuring(1),
                associating(2),
                active(3),
                delay(4)
        }
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "Describes the NCP state of operation:
                        down          – node administratively idle.
                        configuring – es−es topology configuration or
                                          initial startup.
                        associating – handoff in process.
                        active        – other normal operation.
                        delay         – node is in a delay mode, e.g.
                                          remote node waiting between
                 updates."
        ::= { status 1 }

statAdminState OBJECT−TYPE
        SYNTAX INTEGER {
                idle(0),
                active(1)
        }
        MAX-ACCESS read-write
        STATUS current
        DESCRIPTION
                "Indicates whether the process is in an idle mode
                 or active. Setting this object to idle causes the
                 NCP process to remain alive, but stop processing
                 packets. A transition setting from idle to active
                 causes the NCP process to re-initialize as though
                 just started."
        ::= { status 2 }

statTopologyTime OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The time in micro-seconds to do the last optimal
                 topology calculation. A zero value means that
                 the topology calculation has not been performed
                 yet."
        ::= { status 3 }

statBeamformTime OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The time in micro-seconds to do the last beamform.
                 A zero values indicates that beamforming has not
                 been performed yet."
        ::= { status 4 }

statNcpTransferTime OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The time in micro-seconds of the last NCP packet
                 transmission. A zero value means that no packet
                 measurements are being taken."
        ::= { status 5 }

--
-- The VNC-NCP Group
--

vncNcp OBJECT IDENTIFIER ::= { rdrnNcpMIB 12 }

vncNcpHandoff OBJECT IDENTIFIER ::= { vncNcp 1 }

vncNcpHandoffTable OBJECT-TYPE
```

```
        SYNTAX SEQUENCE OF VncNcpHandoffEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "VNC Handoff Prediction Table."
        ::= { vncNcpHandoff 1 }

vncNcpHandoffEntry OBJECT−TYPE
        SYNTAX VncNcpHandoffEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "VNC Handoff Prediction Table."
        INDEX { vncNcpHandoffTime }
        ::= { vncNcpHandoffTable 1 }

VncNcpHandoffEntry ::= SEQUENCE {
        vncNcpHandoffProb
                INTEGER,
        vncNcpHandoffTime
                TimeTicks,
        vncNcpHandoffName
                DisplayString
        }

vncNcpHandoffProb OBJECT−TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "This is the probability that this new ES−RN association
                 will occur."
        ::= { vncNcpHandoffEntry 1 }

vncNcpHandoffTime OBJECT−TYPE
        SYNTAX TimeTicks
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "This is the predicted time of the new ES−RN association."
        ::= { vncNcpHandoffEntry 2 }

vncNcpHandoffName OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "This is the name of the new node predicted to be associated
                 with the current node. This should point to the remoteNode
                 table already defined but with predicted information."
        ::= { vncNcpHandoffEntry 3 }
```

```
vncNcpBeamform OBJECT IDENTIFIER ::= { vncNcp 2 }

vncNcpBeamformTable OBJECT−TYPE
        SYNTAX SEQUENCE OF VncNcpBeamformEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "VNC Beamform prediction table."
        ::= { vncNcpBeamform 1 }

vncNcpBeamformEntry OBJECT−TYPE
        SYNTAX VncNcpBeamformEntry
        MAX−ACCESS not−accessible
        STATUS current
        DESCRIPTION
                "VNC Beamform prediction table."
        INDEX { vncNcpBeamformTime }
        ::= { vncNcpBeamformTable 1 }

VncNcpBeamformEntry ::= SEQUENCE {
        vncNcpBeamformProb
                INTEGER,
        vncNcpBeamformTime
                TimeTicks,
        vncNcpBeamformAngle
                DisplayString
        }

vncNcpBeamformProb OBJECT−TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "This is the probability that this beam information
                 will be valid."
        ::= { vncNcpBeamformEntry 1 }

vncNcpBeamformTime OBJECT−TYPE
        SYNTAX TimeTicks
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "This is the predicted time of this new beam
                 configuration will be valid."
        ::= { vncNcpBeamformEntry 2 }

vncNcpBeamformAngle OBJECT−TYPE
        SYNTAX DisplayString
        MAX−ACCESS read−only
        STATUS current
        DESCRIPTION
                "This is the predicted angle of the beam. Should point
                 to beamtable already defined but with predicted values."
        ::= { vncNcpBeamformEntry 3 }
END
```

The following SNMP MIB is part of the RDRN NCP and LMD. This MIB contains the VNC specific portion of a process and is intended to be generic enough for use by any process using the VNC algorithm. It is a table of the most important parameters of the VNC algorithm. Currently, the remote node GPS, remote node NCP/LMD, and the edge switch have been implemented as LPs with this MIB. When the RN and ES LMDs are running, live data from this table can be viewed from http://www.tisl.ukans.edu/˜sbush/rdrn/ncp.html.

```
RDRN−VNC−MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE−IDENTITY, OBJECT−TYPE, experimental,
    Counter32, TimeTicks
        FROM SNMPv2−SMI
    DisplayString
        FROM SNMPv2−TC;

rdrnVncMIB MODULE−IDENTITY
     LAST−UPDATED "9701010000Z"
     ORGANIZATION "KU TISL"
     CONTACT−INFO
              "Steve Bush sbushtisl.ukans.edu"
     DESCRIPTION
             "Experimental MIB modules for the Rapidly Deployable
              Radio Network (RDRN) Project Network Control
              Protocol (NCP) enhanced with the Virtual Network
              Configuration."
     ::= { experimental ncp(75) 4 }

−−
−− Logical Process Table
−−

logicalProcess OBJECT IDENTIFIER ::= { rdrnVncMIB(4) 1 }

logicalProcessTable OBJECT−TYPE
        SYNTAX SEQUENCE OF LogicalProcessEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "Table of VNC LP information."
        ::= { logicalProcess 1 }

logicalProcessEntry OBJECT-TYPE
        SYNTAX LogicalProcessEntry
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
                "Table of VNC LP information."
        INDEX { logicalProcessID }
        ::= { logicalProcessTable 1 }

LogicalProcessEntry ::= SEQUENCE {
        logicalProcessID
                DisplayString,
        logicalProcessLVT
                INTEGER,
        logicalProcessQRSize
                INTEGER,
        logicalProcessQSSize
                INTEGER,
        logicalProcessRollbacks
                INTEGER,
        logicalProcessSQSize
                INTEGER,
        logicalProcessTolerance
                INTEGER,
        logicalProcessGVT
                INTEGER,
        logicalProcessLookAhead
                INTEGER,
        logicalProcessGvtUpdate
                INTEGER,
        logicalProcessStepSize
                INTEGER
        }

logicalProcessID OBJECT-TYPE
        SYNTAX DisplayString
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "The LP identifier."
        ::= { logicalProcessEntry 1 }

logicalProcessLVT OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is the LP Local Virtual Time."
        ::= { logicalProcessEntry 2 }

logicalProcessQRSize OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is the LP Receive Queue Size."
        ::= { logicalProcessEntry 3 }

logicalProcessQSSize OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is the LP send queue size."
        ::= { logicalProcessEntry 4 }

logicalProcessRollbacks OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is the number of rollbacks this LP has suffered."
        ::= { logicalProcessEntry 5 }

logicalProcessSQSize OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is the LP state queue size."
        ::= { logicalProcessEntry 6 }

logicalProcessTolerance OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is the allowable deviation between process's
                 predicted state and the actual state."
        ::= { logicalProcessEntry 7 }

logicalProcessGVT   OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
```

```
                "This is this system's notion of Global Virtual Time."
        ::= { logicalProcessEntry 8 }

logicalProcessLookAhead  OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is this system's maximum time into which it can
                 predict."
        ::= { logicalProcessEntry 9 }

logicalProcessGvtUpdate OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is the GVT update rate."
        ::= { logicalProcessEntry 10 }

logicalProcessStepSize OBJECT-TYPE
        SYNTAX INTEGER (0..2147483647)
        MAX-ACCESS read-only
        STATUS current
        DESCRIPTION
                "This is the time until next virtual message."
        ::= { logicalProcessEntry 11 }
END
```

Appendix C

Acronyms

C.1 Acronyms

ABR Available Bit Rate. ABR is an ATM layer service category for which the limiting ATM layer transfer characteristics provided by the network may change subsequent to connection establishment. A flow control mechanism is specified which supports several types of feedback to control the source rate in response to changing ATM layer transfer characteristics. It is expected that an end-system that adapts its traffic in accordance with the feedback will experience a low cell loss ratio and obtain a fair share of the available bandwidth according to a network specific allocation policy. Cell delay variation is not controlled in this service, although admitted cells are not delayed unnecessarily.

AHDLC Adaptive High Level Data Link Protocol. An Adaptive HDLC-like link layer protocol used in the Rapidly Deployable Radio Network (RDRN). The frame size and retry mechanism are adaptive.

AMPS Advanced Mobile Phone System. Advanced Mobile Phone System [28] is the North American analog cellular phone standard. Advanced Mobile Phone System operates in the 800 MHz and 900 MHz bands. About 85% of Advanced Mobile Phone System subscribers are in the U.S.

ANO Attach New to Old. The node where old and new connections meet after a Handoff (HO) is called the Attach New to Old point. A term defined in [41] for mobile wireless Asynchronous Transfer Mode (ATM) architecture along with Mobile Terminal (MT), Base Transceiver Station (BTS), Handoff (HO), and Cell Site Switch (CSS). These elements are shown in Figure 2.1.

ARP Address Resolution Protocol. A TCP/IP Protocol that dynamically binds a Network Layer IP address to a Data Link Layer physical hardware address.

ATMARP Asynchronous Transfer Mode Address Resolution Protocol. A TCP/IP Protocol for dynamically binding a Network Layer IP address to an ATM address. Described in [49].

ATM Asynchronous Transfer Mode. A transfer mode in which the information is organized into cells. It is asynchronous in the sense that the recurrence of cells containing information from an individual user is not necessarily periodic.

BTS Base Transceiver Station. A non-mobile base station which has both wired and wireless Asynchronous Transfer Mode (ATM) ports. A term defined in [41] for mobile wireless Asynchronous Transfer Mode (ATM) architecture along with Mobile Terminal (MT), Attach New to Old (ANO), Handoff (HO), and Cell Site Switch (CSS). These elements are shown in Figure 2.1.

C/E Condition Event Network. A Condition Event Network network consists of state and transition elements which contain tokens. Tokens reside in state elements. When all state elements leading to a transition element contain a token, several changes take place in the Condition Event Network network. First, the tokens are removed from the conditions which triggered the event, the event occurs, and finally tokens are placed in all state outputs from the transition which was triggered. Multiple tokens in a condition and the uniqueness of the tokens is irrelevant in a Condition Event Network Net.

CDMA Code Division Multiple Access. A coding scheme, used as a modulation technique, in which multiple channels are independently coded for transmission over a single wide-band channel. Note 1: In some communication systems, CDMA is used as an access method that permits carriers from different stations to use the same transmission equipment by using a wider bandwidth than the individual carriers. On reception, each carrier can be distinguished from the others by means of a specific modulation code, thereby allowing for the reception of signals that were originally overlapping in frequency and time. Thus, several transmissions can occur simultaneously within the same bandwidth, with the mutual interference reduced by the degree of orthogonality of the unique codes used in each transmission. Note 2: CDMA permits a more uniform distribution of energy in the emitted bandwidth.

CDPD Cellular Digital Packet Data. Cellular Digital Packet Data is described in [20] and [61] protocol. The Cellular Digital Packet Data protocol operates much like the Mobile IP protocol for mobile cellular voice networks and is designed to co-exist with the Advanced Mobile Phone System[28]. Cellular Digital Packet Data uses registration and encapsulation to forward packets to the current location of the mobile host just as in Mobile-IP. Cellular Digital Packet Data was developed independently of Mobile-IP.

CE Clustered Environment. One of the contributions of [5] in Clustered Time Warp is an attempt to efficiently control a cluster of Logical Processs on a processor by means of the Clustered Environment. The Clustered Environment allows multiple Logical Process (LP) to behave as individual Logical Processs as in the basic time warp algorithm or as a single collective Logical Process.

CMB Chandy-Misra-Bryant. A Conservative distributed simulation synchronization technique.

CMIP Common Management Information Protocol. A protocol used by an application process to exchange information and commands for the purpose of managing remote computer and communications resources. Described in [50].

CRC Cyclic Redundancy Checksum. An error-detection scheme that (a) uses parity bits generated by polynomial encoding of digital signals, (b) appends those parity bits to the digital signal, and (c) uses decoding algorithms that detect errors in the received digital signal. Note: Error correction, if required, may be accomplished through the use of an automatic repeat-request (ARQ) system.

CSS Cell Site Switch. A wired Asynchronous Transfer Mode (ATM) switch near the Base Transceiver Station (BTS). A term defined in [41] for mobile wireless Asynchronous Transfer Mode (ATM) architecture along with Mobile Terminal (MT), Attach New to Old (ANO), Handoff (HO), and Base Transceiver Station (BTS). These elements are shown in Figure 2.1.

CS Current State. The current value of all information concerning a Physical Process (PP) encapsulated by a Logical Process (LP) and all the structures associated with the Logical Process.

CTW Clustered Time Warp. Clustered Time Warp is an optimistic distributed simulation mechanism described in [5].

DES Data Encryption Standard. A cryptographic algorithm for the protection of unclassified computer data and published by the National Institute of Standards and Technology in Federal Information Processing Standard Publication 46-1. Note: DES is not approved for protection of national security classified information.

DHCP Dynamic Host Configuration Protocol. The Dynamic Host Configuration Protocol [27] provides for automatic configuration of a host from data stored on the network.

EN Edge Node. A Rapidly Deployable Radio Network (RDRN) [16] node which provides connectivity between a wired and wireless Asynchronous Transfer Mode (ATM) network.

ES Edge Switch. A component of the Rapidly Deployable Radio Network (RDRN) [16] Edge Node (EN) which contains internal Asynchronous Transfer Mode (ATM) switching capability for wireless input ports.

ESN Electronic Serial NumberṪhe Electronic Serial Number is stored along with the Mobile Identification Number (MIN) in the Home Location Register (HLR) in the Advanced Mobile Phone System (AMPS) [28].

FH Fixed Host. A host directly connected to a wired network.

FSM Finite State Machine. A five-tuple consisting of a set of states, an input alphabet, an output alphabet, a next-state transition function, and an output function. Used to formally describe the operation of a protocol.

GFC Generic Flow Control. Generic Flow Control is a field in the ATM header which can be used to provide local functions (e.g., flow control). It has local significance only and the value encoded in the field is not carried end-to-end.

GPS Global Positioning System. A satellite-based positioning service developed and operated by the Department of Defense.

GSM Global System for Mobile Communication. A pan-European standard for digital mobile telephony which provides a much higher capacity than traditional analog telephones as well as diversified services (voice, data) and a greater transmission security through information encoding for users across Europe. Global System for Mobile Communication is described in [31].

GSV Global Synchronic Distance. The maximum Synchronic Distance in a Petri-Net model of a system.

GVT Global Virtual Time. The largest time beyond which a rollback based system will never rollback.

HDLC High Level Data Link Protocol. A link-level protocol used to facilitate reliable point to point transmission of a data packet. A subset of HDLC, known as LAP-B is the Layer two protocol for CCITT Recommendation X.25.

HLR Home Location Register. The Home Location Register and Visiting Location Register are used in mobile cellular voice networks [28] and [31]. These agents maintain the information concerning whether a mobile node is currently associated with its home network or is a visiting mobile node, that is, not currently in its home network.

HO Handoff. A change in association of a mobile node from one base station to another. A term defined in [41] for mobile wireless Asynchronous Transfer Mode (ATM) architecture along with Mobile Terminal (MT), Attach New to Old (ANO), Handoff (HO), and Base Transceiver Station (BTS). These elements are shown in Figure 2.1.

IETF Internet Engineering Task Force. The main standards organization for the Internet. The IETF is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. It is open to any interested individual.

IPC Inter-Processor Communication. Communication among Unix processes. This may take place via sockets, shared memory, or semaphores.

LIS Logical IP Subnet. In the LIS scenario described in [49], each separate administrative entity configures its hosts and routers within a closed logical IP subnetwork. Each LIS operates and communicates independently of other acLISs on the same ATM network. Hosts connected to ATM communicate directly to other hosts within the same LIS. Communication to hosts outside of the local LIS is provided via an IP router. This router is an ATM Endpoint attached to the ATM network that is configured as a member of one or more LISs. This configuration may result in a number of disjoint LISs operating over the same ATM network. Hosts of differing IP subnets MUST communicate via an intermediate IP router even though it may be possible to open a direct VC between the two IP members over the ATM network.

LN Logical Node. A logical node that represents a lower level peer group as a single point for purposes of operating at one level of the PNNI routing hierarchy.

LP Logical Process. An Logical Process consists of the Physical Process (PP) and additional data structures and instructions which maintain message order and correct operation as a system executes ahead of real time.

LVT Local Virtual Time. The Logical Process (LP) contains its notion of time known as Local Virtual Time (LVT).

MAC Media Access Control. The part of the data link layer that supports topology-dependent functions and uses the services of the physical layer to provide services to the logical link control sub-layer.

MH Mobile Host. A term used to describe a mobile end system in [99].

MIB Management Information Base. A collection of objects which can be accessed by a network management protocol.

MIN Mobile Identification Number. The Mobile Identification Number is stored along with the Electronic Serial Number (ESN) in the Home Location Register (HLR) in the Advanced Mobile Phone System (AMPS) [28].

MSC Mobile Switching Center. The Home Location Register and Visiting Location Register reside on the Mobile Switching Center which is connected to each mobile base station in the Advanced Mobile Phone System (AMPS). The Mobile Switching Center controls call setup, call transfer, billing, interaction with the PSTN, and other functions using Signaling System 7 (SS7).

MSR Mobile Support Router. The mobile TCP/IP known as I-TCP requires Mobile Support Router which have the ability to transfer images of sockets involved in an established TCP connection from one MSR to another.

MTW Moving Time Windows. Moving Time Windows is a distributed simulation algorithm that controls the amount of aggressiveness in the system by means of a moving time window Moving Time Windows. The trade-off in having no rollbacks in this algorithm is loss of fidelity in the simulation results.

MT Mobile Terminal. A mobile host end system which receives wireless Asynchronous Transfer Mode (ATM) cells. A term defined in [41] for mobile wireless Asynchronous Transfer Mode (ATM) architecture along with Cell Site Switch (CSS), Attach New to Old (ANO), Handoff (HO), and Base Transceiver Station (BTS). These elements are shown in Figure 2.1.

NCP Network Control Protocol. The Rapidly Deployable Radio Network (RDRN) protocol which supports wireless Asynchronous Transfer Mode (ATM) communications.

NFT No False Time-stamps. No False Time-stamps Time Warp assumes that if an incorrect computation produces and incorrect event ($E_{i,T}$), then it must be the case that the correct computation also produces an event ($E_{i,T}$) with the same timestamp This simplification makes the analysis in [35] tractable.

NHRP Next Hop Resolution Protocol. The Next Hop Resolution Protocol [95] allows Asynchronous Transfer Mode Address Resolution Protocol to take place across Logical IP Subnet (LIS).

NPSI Near Perfect State Information. The Near Perfect State Information Adaptive Synchronization Algorithms for PDES are discussed in [104] and [103]. The adaptive algorithms use feedback from the simulation itself in order to adapt. The NPSI system requires an overlay system to return feedback information to the Logical Processs. The Near Perfect State Information Adaptive Synchronization Algorithm examines the system state (or an approximation of the state) calculates an error potential for future error, then translates the error potential into a value which controls the amount of optimism.

NTP Network Time Protocol. A TCP/IP time synchronization mechanism. Network Time Protocol [82] is not required in Virtual Network Configuration (VNC) on the Rapidly Deployable Radio Network (RDRN) because each host in the Rapidly Deployable Radio Network network has its own Global Positioning System receiver.

PA Perturbation Analysis. The technique of Perturbation Analysis allows a great deal more information to be obtained from a single simulation execution than explicitly collected statistics. It is particularly useful for finding the sensitivity information of simulation parameters from the sample path of a single simulation run. May be an ideal way for VNC to automatically adjust tolerances and provide feedback to driving process(es). Briefly, assume a sample path, (Θ, ξ) from a simulation. Θ is vector of all parameters ξ is vector of all random occurrences. $L(\Theta, \xi)$ is the sample performance. $J(\Theta)$ is the average performance, $E[L(\Theta, \xi)]$. Parameter changes cause perturbations in event timing. Perturbations in event timing propagate to other events. This induces perturbations in L. If perturbations into (Θ, ξ) are small, assume event trace $(\Theta + d\Theta, \xi)$ remains unchanged. Then $\frac{dL(\Theta,\xi)}{d\Theta}$ can be calculated. From this, the gradient of $J(\Theta)$ can be obtained which provides the sensitivity of performance to parameter changes. PA can be used to adjust tolerances while VNC is executing because event times are readily available in the State Queue (SQ).

PBS Portable Base Station. Portable Base Station nodes are connected via Virtual Path Trees which are part of the wireless Asynchronous Transfer Mode BAHAMA [29, 56, 116] network.

PCN Personal Communications Network. A set of capabilities that allows some combination of terminal mobility, personal mobility, and service profile management. Note 1: The flexibility offered by PCS can supplement existing telecommunications services, such as cellular radio, used for NS/EP missions. Note 2: PCS and UPT are sometimes mistakenly assumed to be the same service concept. UPT allows complete personal mobility across multiple networks and service providers. PCS may use UPT concepts to improve subscriber mobility in allowing roaming to different service providers, but UPT and PCS are not the same service concept. Contrast with Universal Personal Telecommunications service.

PDES Parallel Discrete Event Simulation. Parallel Discrete Event Simulation is a class of simulation algorithms which partition a simulation into individual events and synchronizes the time the events are executed on multiple processors such that the real time to execute the simulation is as fast as possible.

PDU Protocol Data Unit. 1. Information that is delivered as a unit among peer entities of a network and that may contain control information, address information, or data. 2. In layered systems, a unit of data that is specified in a protocol of a given layer and that consists of protocol-control information of the given layer and possibly user data of that layer.

PGL Peer Group Leader. A node which has been elected to perform some of the functions associated with a logical group node.

PG Peer Group. A set of logical nodes which are grouped for purposes of creating a routing hierarchy. PTSEs are exchanged among all members of the group.

P/T Place Transition Net. A P/T Network is exactly like a Condition Event Network (C/E) Net except that a P/T Net allows multiple tokens in a place and multiple tokens may be required to cause a transition to fire.

PIPS Partially Implemented Performance Specification. Partially Implemented Performance Specification is a hybrid simulation and real-time system which is described in [7]. Components of a performance specification for a distributed system are implemented while the remainder of the system is simulated. More components are implemented and tested with the simulated system in an iterative manner until the entire distributed system is implemented.

PNNI Private Network-Network Interface. A routing information protocol that enables extremely scalable, full function, dynamic multi-vendor ATM switches to be integrated in the same network.

PP Physical Process. A Physical Process is nothing more than an executing task defined by program code. An example of a Physical Process is the Rapidly Deployable Radio Network (RDRN) beam table creation task. The beam table creation task generates a table of complex weights which controls the angle of the radio beams based on position input.

Q.2931 Q.2931. Q.2931 is the ITU standard for Asynchronous Transfer Mode (ATM) signaling described in [19].

QR Receive Queue. A queue used in the Virtual Network Configuration (VNC) algorithm to hold incoming messages to a Logical Process (LP). The messages are stored in the queue in order by receive time.

QS Send Queue. A queue used in the Virtual Network Configuration (VNC) algorithm to hold copies of messages which have been sent by a Logical Process (LP). The messages in the Send Queue (QS) may be sent as anti-messages if a rollback occurs.

QoS Quality of Service. Quality of Service is defined on an end-to-end basis in terms of the following attributes of the end-to-end ATM connection: Cell Loss Ratio, Cell Transfer Delay, Cell Delay Variation.

RDRN Rapidly Deployable Radio Network. This project, funded by the Information Technology Office (ITO) of the Advanced Research Projects Agency, involves the design and implementation of a reconfigurable ATM wireless network which uses antenna beamforming for improved spatial reuse of frequencies. The RDRN project is associated with the ARPA Global Mobile Information Systems initiative.

RN Remote Node. A host station end system in the Rapidly Deployable Radio Network (RDRN).

RT Real Time. The current wall clock time.

SID Station Identification. The Station Identification is used in the Advanced Mobile Phone System (AMPS) [28] to identify a base station.

SLP Single Processor Logical Process. Multiple Logical Process (LP) executing on a single processor.

SLW Sliding Lookahead Window. The Sliding Lookahead Window is used in Virtual Network Configuration (VNC) to limit or throttle the prediction rate of the Virtual Network Configuration system. The Sliding Lookahead Window is defined as the maximum time into the future for which the acVNC system may predict events.

SNMP Simple Network Management Protocol. The Transmission Control Protocol/Internet Protocol (TCP/IP) standard protocol that (a) is used to manage and control IP gateways and the networks to which they are attached, (b) uses IP directly, bypassing the masking effects of TCP error correction, (c) has direct access to IP datagrams on a network that may be operating abnormally, thus requiring management, (d) defines a set of variables that the gateway must store, and (e) specifies that all control operations on the gateway are a side-effect of fetching or storing those data variables, i.e., operations that are analogous to writing commands and reading status. Simple Network Management Protocol is described in [94].

SQ State Queue. The State Queue is used in Virtual Network Configuration (VNC) as a Logical Process (LP) structure to hold saved state information for use in case of a rollback. The State Queue is the cache into which pre-computed results are stored.

SS7 Signaling System 7Å common-channel signaling system defined by the CCITT in the 1988 Blue Book, in Recommendations Q.771 through Q.774. Note: SS7 is a prerequisite for implementation of an Integrated Services Digital Network (ISDN).

TDMA Time Division Multiple Access. A communications technique that uses a common channel (multi-point or broadcast) for communications among multiple users by allocating unique time slots to different users. Note: TDMA is used extensively in satellite systems, local area networks, physical security systems, and combat-net radio systems.

TDN Temporary Directory Number. If the visiting mobile node can be reached in the Advanced Mobile Phone System (AMPS), the visiting mobile's Mobile Switching Center (MSC) will respond with a Temporary Directory Number.

TNC Terminal Node Controller. The Terminal Node Controller automatically divides the message into packets, keys the transmitter and sends the packets. While receiving packets, the TNC automatically decodes, checks for errors, and displays the received messages. In addition, any packet TNC can be used a packet relay station, sometimes called a digipeater. This allows for greater range by stringing several packet stations together.

TOE Time of Expiry. A priority queuing mechanism proposed in [91] for queuing wireless Asynchronous Transfer Mode (ATM) cells. Call request packets indicate an expiry time for data to be transmitted. Cells are then served in order of expiry with the smallest expiry time served first. The Time of Expiry mechanism is proposed to reduce packet loss for real-time data.

TR Receive Time. The time a Virtual Network Configuration (VNC) message value is predicted to be valid.

TS Send Time. The Local Virtual Time (LVT) that a virtual message has been sent. This value is carried within the header of the message. The Send Time is used for canceling the effects of false messages.

VC Virtual Circuit. A communications channel that provides for the sequential unidirectional transport of ATM cells.

VCI Virtual Circuit Identifier. Virtual Channel Identifier: A unique numerical tag as defined by a 16 bit field in the ATM cell header that identifies a virtual channel, over which the cell is to travel.

VLR Visiting Location Register. The Visiting Location Register and Home Location Register are used in mobile cellular voice networks [28] and [31]. These agents maintain the information concerning whether a mobile node is currently associated with its home network or is a visiting mobile node, that is, not currently in its home network.

VNC Virtual Network Configuration. Virtual Network Configuration allows future states of a system to be predicted and used efficiently via optimistic distributed simulation technique applied to a real time system.

VP Virtual Path. A unidirectional logical association or bundle of Virtual Circuit (VC)s.

VPI Virtual Path Identifier. An eight bit field in the ATM cell header which indicates the Virtual Path (VP) over which the cell should be routed.

VTRP Virtual Trees Routing Protocol. A wireless Asynchronous Transfer Mode routing protocol used in the BAHAMA [29, 56, 116] network. Portable Base Station (PBS) nodes are connected via Virtual Path Trees which are preconfigured Asynchronous Transfer Mode (ATM) Virtual Path (VP)s. These trees can change based on the topology as described in the *Virtual Trees Routing Protocol* [57].

Appendix D

Network Management Simulation Architecture

The simulation has been implemented with Maisie [6]. Its suitability for this has been demonstrated in the Rapidly Deployable Radio Network network management and control design and development and in [100] to develop a mobile wireless network parallel simulation environment. The parallel simulation environment shows a speedup over the currently used commercial sequential simulation packages. The environment and a set of modules which have been developed for mobile network simulation are described in [100]. Maisie uses a language which has been influenced by a classic work describing the characteristics of a parallel programming language structure [44]. The programming features developed here are used in many parallel programming languages besides Maisie. Since every Maisie entity has a built-in input queue, each Logical Process is comprised of three additional Maisie entities:

- An entity which represents the Physical Process

- An entity for the Logical Process state queue

- An entity for the Logical Process output message queue

There is also a gvt entity for the calculation of Global Virtual Time (Global Virtual Time). All three of the above entities work together to implement Virtual Time as described in [54]. The first entity above, representing the Physical Process, contains a delay mechanism in order to implement the sliding lookahead window. The gvt process should notify all processes to cease forward simulation when Global Virtual Time reaches the end of the window. In this version of Virtual Network Configuration, each Logical Process simply compares its LVT to the current time and holds processing until current time is back within the lookahead sliding window.

Determination of Global Virtual Time (Global Virtual Time) should be done as defined by [63]. This algorithm allows Global Virtual Time to be determined in a message-passing environment as opposed to the easier case of a shared memory environment. It also allows normal processing to continue during the Global Virtual Time determination phase. However, in this implementation each output message is sent to the gvt entity as well as to its proper destination. In addition, the gvt entity checks all Logical Processs for their current LVT and chooses the minimum message send time and LVT as the current Global Virtual Time. The gvt entity is allowed to execute in parallel with the other entities in this simulation, it does not stop the other entities while performing its computation and thus may not always be perfectly accurate. However, the results were close enough for the purpose of these experiments.

State adjustment rollbacks are the most critical part of Virtual Network Configuration. They are handled in a slightly different fashion from causality failure rollbacks. A state verification failure causes the Logical Process state to be corrected at the time of the state verification which failed. The state, S_v, has been obtained from the actual device from the state adjustment query at time t_v. The Logical Process rolls back to exactly t_v with state, S_v. States greater than t_v are removed from the state queue. Anti-messages are sent from the output message queue for all messages greater than t_v. The Logical Process continues forward execution from this point. Note that this implies that the message and state queues cannot be purged of elements which are older than the Global Virtual Time. Only elements which are older than real time can be purged.

Appendix E

Complexity in Self-Predictive Systems

A fascinating perspective on the topic of self-predictive systems is found in *Gödel, Escher, and Bach: An Eternal Golden Braid* which is a wonderful look at the nature of Human and Artificial Intelligence. A central point in [45] is that intelligence is a Tangled Hierarchy, illustrated in Figure E.1. A hand performing the act of drawing is expected to be in a level above the hand being drawn. When the two levels are folded together a Tangled Hierarchy results, which is expressed much more elegantly in [45]. Virtual Network Configuration as presented in this work is a Tangled Hierarchy on several levels: simulation-reality and also present-future time. One of the hands in Figure E.1 represents prediction based on simulation and the other represents reality, each modifying the other in the Virtual Network Configuration algorithm. However, there is a much deeper mathematical relationship present in this algorithm which relates to Gödel's Theorem. In a nutshell, Gödel's theorem states that no formal system can describe itself with complete fidelity. This places a fundamental limitation on the ability of mathematics to describe itself. The implication to artificial intelligence is that the human mind can never fully understand its own operation, or possibly that if one could fully understand how one thinks while one is thinking, then one would cease to "be". In the much more mundane Virtual Network Configuration algorithm, a system is in some sense attempting to use itself to predict its own future state with the goal of perfect fidelity. If Gödel's Theorem applies, then perfect fidelity is an impossible goal. However, by allowing for a given tolerance in the amount of error and assuming accuracy in prediction which increases as real time approaches the actual time of an event, this study assumes that a useful self-predictive system can be implemented.

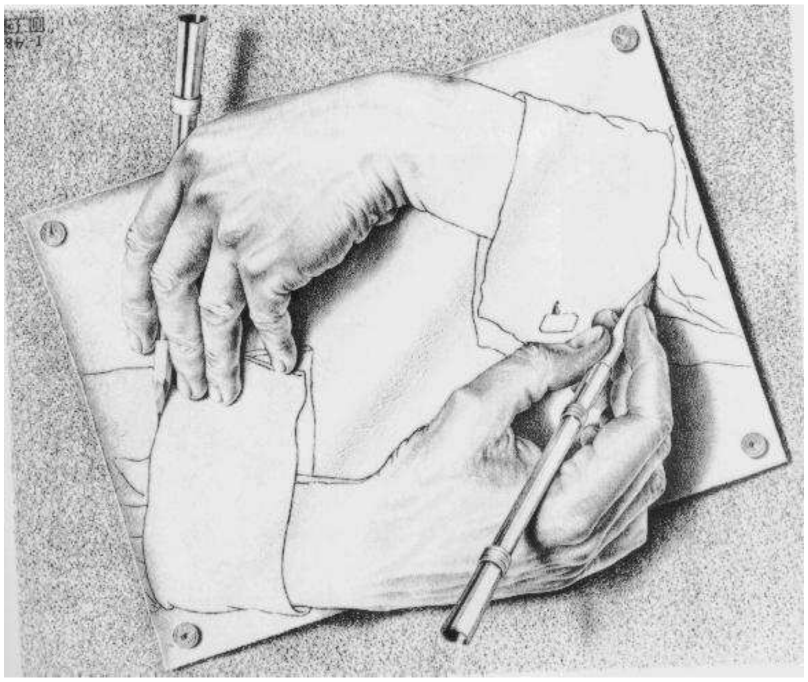


Figure E.1: A Tangled Hierarchy. *Drawing by M. C. Escher*.

Bibliography

[1] Anthony S. Acampora and Mahmoud Naghshineh. An Architecture and Methodology for Mobile-Executed Handoff in Cellular ATM Networks. *IEEE Journal of Selected Areas in Communication*, 12(8), October 1994.

[2] Active Networks Group. *Active Network Encapsulation Protocol (ANEP)*, July 1997.

[3] Pramthima Agrawal, Eoin Hyden, Paul Kryzanowski, Partho Mishrea, Mani B. Srivastava, and John A. Trotter. SWAN: A Mobile Multimedia Wireless Network. *IEEE Personal Communications*, April 1996.

[4] Herve Avril. *Clustered Time Warp and Logic Simulation*. PhD thesis, McGill University, December 1996.

[5] Herve Avril and Carl Tropper. Clustered Time Warp and Logic Simulation, 1995.

[6] Rajive Bagrodia and Wen-Toh Liao. *Maisie User Manual Release 2.1*, jun 1993.

[7] Rajive Bagrodia and Chien-Chung Shen. MIDAS: Integrated Design and Simulation of Distributed Systems. *IEEE Transactions on Software Engineering*, October 1991.

[8] Ajay Bakre and B. R. Badrinath. I-TCP: Indirect TCP for Mobile Hosts. Technical Report DCS-TR-314, Rutgers Dept of Computer Science, October 1994.

[9] Duane Ball and Susan Hoyt. The Adaptive Time-Warp Concurrency Control Algorithm. In *Proceeding of SCS'90*, 1990.

[10] Oran Berry and David Jefferson. Critical Path Analysis of Distributed Simulation. In *SCS Multi Conference on Distributed Simulation*, 1985.

[11] Samrat Bhattacharjee, Kenneth L. Calvert, and Ellen W. Zegura. On active networing and congestion. Technical Report GIT-CC-06/02, Geogia Institute of Technology, 1996.

[12] A. Boukerche and C. Tropper. A Static Partitioning and Mapping Algorithm for Conservative Parallel Simulations. In *Proceedings of the 8th Workshop on Parallel and Distributed Simulation*, pages 164,172, 1994.

[13] Stephen F. Bush, Joseph B. Evans, and Victor Frost. Mobile ATM Buffer Capacity Analysis. *ACM-Baltzer Mobile Networks and Nomadic Applications (NOMAD)*, 1(1), February 1996. URL: http://www.ittc.ukans.edu/∼sbush.

[14] Stephen F. Bush, Victor S. Frost, and Joseph B. Evans. Network Management of Predictive Mobile Networks. *Journal of Network and Systems Management*, 1999. To be published in *Journal of Network and Systems Management*, URL: http://www.ittc.ukans.edu/∼sbush.

[15] Stephen F. Bush, Sunil Jagannath, Joseph B. Evans, and Victor Frost. A Control and Management Network for Wireless ATM Systems. In *Proceedings of the International Communications Conference '96*, pages 459,463, June 1996. URL: http://www-ee.uta.edu/organizations/commsoc/commsoft.html.

[16] Stephen F. Bush, Sunil Jagannath, Joseph B. Evans, Victor Frost, Gary Minden, and K. Sam Shanmugan. A Control and Management Network for Wireless ATM Systems. *ACM-Baltzer Wireless Networks (WINET)*, 3:267,283, 1997. URL: http://www.ittc.ukans.edu/∼sbush.

[17] Stephen F. Bush, Sunil Jagannath, Ricardo Sanchez, Joseph B. Evans, Victor Frost, and K. Sam Shanmugan. Rapidly Deployable Radio Networks (RDRN) Network Architecture. Technical Report 10920-09, Telecommunications & Information Sciences Laboratory, July 1995. URL: http://www.tisl.ukans.edu/-∼sbush .

[18] Ramon Caceres and Liviu Iftode. Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. *IEEE Journal on Selected Areas in Communications*, 13(5), June 1995.

[19] CCITT. *Q.2931*, 1995. Online version available at gopher://cell-relay.indiana.edu/11/docs/current/CCITT/Q.2931.

[20] CDPD Forum, 1996. http://www.cdpd.org.

[21] K. M. Chandy and J. Misra. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering*, September 1979.

[22] Rao Cherukuri and Doug Dykeman, editors. *ATM Forum 94-0471R13 PNNI Draft Specification*. ATM Forum, 1994.

[23] J. H. Condon, T. S. Duff, M. F. Jukl, C. R. Kalmanek, B. N. Locanthi, J. P. Savicki, and J. H. Venutolo. Rednet: A Wireless ATM Local Area Network using Infrared Links. In *Mobicom '95*, pages 151,159, 1995.

[24] Vince Darely. Emergent Phenomena and Complexity. Technical report, Division of Applied Sciences, Harvard University, 1996.

[25] Martin de Prycker. *Asynchronous Transfer Mode*. Ellis Horwood, 1993. ISBN: 0-13-053513-3.

[26] Antonio DeSimone, Mooi Choo Chuah, and On-Ching Yue. Throughput Performance of Transport-Layer Protocols over Wireless LANs. In *Proceedings of Globecom '93*, 1993.

[27] R. Droms, editor. *Dynamic Host Configuration Protocol*. Bucknell University, October 1993. Online version available at http://ds.internic.net/rfc/rfc1541.txt.

[28] EIA/TIA. Cellular Radio-telecommunications Intersystem Operations, January 1992.

[29] K. Y. Eng, M. J. Karol, M. Veeraraghavan, E. Ayanoglu, C. B. Woodworth, P. Pancha, and R. A. Valenzuela. BAHAMA: A Broadband Ad-Hoc Wireless ATM Local-Area Network. In *Proceedings of ICC'95*, pages 1216,1223, February 1995.

[30] Ericsson Business Area Radio Communications. *Update 1996: The Wireless Marketplace*, 1996. http://www.ericsson.nl/EPI/BR/TDMA/tdma3.html.

[31] ETSI. Mobile Application Part (MAP) Specification.

[32] Robert E. Felderman and Leonard Kleinrock. An Upper Bound on the Improvement of Asynchronous versus Synchronous Distributed Processing. In *SCS '90*, January 1990.

[33] Analucia Schiaffino Morales De Franceschi, Liuz Fernando Kormann, and Carlos Becker Westphall. Performance Evaluation for Proactive Network Management. In *Proceedings of ICC'96*, pages 22,26, 1996.

[34] Richard M. Fujimoto. Parallel Discrete Event Simulation. *Communications of the ACM*, 33(10):30–53, October 1990.

[35] Kaushik Ghosh, Richard M. Fujimoto, and Karsten Schwan. Time Warp Simulation in Time Constrained Systems. In *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, pages 163,166, 1993.

[36] D. M. Glazer. *Load Balancing Parallel Discrete Event Simulations*. PhD thesis, McGill University, 1993.

[37] D. M. Glazer and C. Tropper. A Dynamic Load Balancing Algorithm for Time Warp. *Parallel and Distributed Systems*, 4(3):318,327, March 1993.

[38] Anurag Gupta, Ian F. Akyldiz, and Richard M. Fujimoto. Performance Analysis of Time Warp with Multiple Homogeneous Processors. *IEEE Transactions on Software Engineering*, October 1991.

[39] Shane Haas. A Consistent Labeling Algorithm for the Frequency/Code Assignment in a Rapidly Deployable Radio Network (RDRN). Technical Report TISL-10920-04, Telecommunications & Information Sciences Laboratory, Jan 1995.

[40] Shane M. Haas, David W. Petr, and John Paden. Edge Node to Remote Node Topology Optimization in the Rapidly Deployable Radio Network (RDRN). Technical Report ITTC-FY97-TR-10920-26, Information and Telecommunications Technology Center, June 1997.

[41] L. Van Hauwermeiren, L. Vercauteren, A. Saidi, and T. Van Landegem. Requirements for Mobility Support in ATM. In *Globecom 94*, pages 1691,1695, 1994.

[42] David Heibeler. The Swarm Simulation System and Individual-based Modeling, 1994. URL: http://www.santefe.edu/swarm.

[43] Yu-Chi Ho. Perturbation Analysis: Concepts and Algorithms. In *Proceedings of the 1992 Winter Simulation Conference*, 1992.

[44] C. A. R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, August 1981.

[45] Douglas R. Hofstadter. *Gödel, Escher, Bach: An Eternal Golden Braid*. Vintage Books, 1980. ISBN 0-394-74502-7.

[46] Daehyoung Hong and Stephen S. Rappaport. Traffic Model and Performance Analysis for Cellular Mobile Radio Telephone Systems with Prioritized and Non prioritized Handoff Procedures. *IEEE Transactions on Vehicular Technology*, Aug 1986.

[47] B. A. Huberman and T. Hogg. The Emergence of Computational Ecologies. In Lynn Nadel and Daniul Stien, editors, *1992 Lectures in Complex Systems*, pages 185,205. Addison-Wesley, 1993.

[48] IEEE. *AX.25 Amateur Packet Radio Link-Layer Protocol*, October 1984.

[49] IETF. *Classical IP and ARP over ATM*, 1995. Online version available at http:/-/ds.internic.net/rfc/rfc1577.txt.

[50] ISO. Open Systems Interconnection - Management Protocol Specification - Part 2: Common Management Information Protocol.

[51] E. Atlee Jackson. Chaos Concepts. In Lynn Nadel and Daniul Stien, editors, *1992 Lectures in Complex Systems*, pages 463,488. Addison-Wesley, 1993.

[52] Van Jacobson. Congestion Avoidence and Control. In *Proc. ACM SIGCOMM '88*, Aug 1988.

[53] Sunil Jagannath. An Adaptive Data Link Layer Protocol for Wireless ATM Networks. Master's thesis, Telecommunications and Information Science Laboratory University of Kansas, June 1997.

[54] D. R. Jefferson and H. A. Sowizral. Fast Concurrent Simulation Using The Time Warp Mechanism, Part I: Local Control. Technical Report TR-83-204, The Rand Corporation, 1982.

[55] V. Jha and R. L. Bagrodia. A Unified Framework for Conservative and Optimistic Distributed Simulation. In *Proceedings of the 8th Workshop on Parallel and Distributed Simulation*, pages 12,19, 1994.

[56] Mark Karol, M. Veerarghavan, and K. Y. Eng. Mobility-Management and Media-Access Issues in the BAHAMA Wireless ATM LAN. In *Proceedings of ICUPC '95*, pages 758,762, 1995.

[57] I. Katzela and M. Veerarghavan. Virtual Trees Routing Protocol for a Wireless ATM LAN. In *IEEE International Conference on Universal Personal Communications*, 1996.

[58] James D. Kittock. Emergent Conventions and the Structure of Multi-agent Systems. In Lynn Nadel and Daniul Stien, editors, *1993 Lectures in Complex Systems*, pages 507,521. Addison-Wesley, 1995.

[59] L. Kleinrock. *Queuing Systems Volume I: Theory*. John Wiley and Sons, 1975.

[60] P. Konas and P. C. Yew. Partitioning for Synchronous Parallel Simulation. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation*, pages 181,184, 1995.

[61] Charles A. Kunzinger. Network Layer Mobility: Comparison of CDPD and Mobile-IP. Technical Report TR 29.2003, Emerging Technologies, 1995.

[62] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, July 1978.

[63] Edward Lazowaska and Yi-Bing Lin. Determining the Global Virtual Time in a Distributed Simulation. Technical Report 90-01-02, University of Washington, 1990.

[64] Ulana Legedza, David J. Wetherall, and John Guttag. Improving the Performance of Distributed Applications Using Active Networks. *Submitted to IEEE INFOCOM*, 1998.

[65] F. Lehmann, R. Seising, and E. Walther-Klaus. Simulation of Learning in Communication Networks. *Simulation Practice and Theory*, 1(1):41–48, July 1993.

[66] Hong Va Leong and Divyakant Agrawal. Semantics-based Time Warp Protocols. In *Proceedings of the 7th International Workshop on Distributed Simulation*, 1994.

[67] Yi-Bing Lin. Understanding the Limits of Optimistic and Conservative Parallel Simulation. Technical Report UWASH-90-08-02, University of Washington, 1990.

[68] Yi-Bing Lin and Edward D. Lazowska. Optimality Considerations of ”Time Warp” Parallel Simulation. Technical Report UWASH-89-07-05, University of Washington, January 1990.

[69] Richard J. Lipton and David W. Mizell. Time Warp vs. Chandy-Misra: A Worst-Case Comparison. In *SCS ’90*, January 1990.

[70] George Liu, Alexander Marlevi, and Gerald Q. Maguire Jr. A Mobile Virtual-Distributed System Architecture for Supporting Wireless Mobile Computing and Communications. In *Mobicom ’95*, 1995.

[71] George Y. Liu. *The Effectiveness of a Full-Mobility Architecture for Wireless Mobile Computing and Personal Communications*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, March 1996.

[72] George Y. Liu and Gerald Q. Maguire Jr. A Predictive Mobility Management Algorithm for Wireless Mobile Computing and Communications. In *International Conference on Universal Personal Communications (ICUPC)*, pages 268,272, Nov 1995.

[73] B. Lubachevsky, A. Schwatz, and A. Weiss. Rollback Sometimes Works ... if Filtered. In *Proceedings of the 1989 Winter Simulation Conference*, pages 630–639, December 1989.

[74] B. D. Lubachevsky. Efficient Distributed Event Driven Simulations of Multiple-Loop Networks. *Communications of the ACM*, 32(1):111–131, 1989.

[75] David G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1989.

[76] A. Lux and D. Steiner. Understanding Cooperation: An Agent's Perspective. In *First International Conference on Multi-agent Systems*, June 1995.

[77] Vijay Madisetti, Jean Walrand, and David Messerschmitt. MTW: Experimental Results for a Constrained Optimistic Scheduling Paradigm. In *Proc. 1987 Winter Simulation Conference*, 1987.

[78] Jeff McAffer. A Unified Distributed Simulation System. In *Proceedings of the 1990 Winter Simulation Conference*, pages 415,422, 1990.

[79] K. McCloghrie, editor. *Management Information Base for Network Management of TCP/IP-based Internets*. Hughes LAN Systems, May 1990. Online version available at http://ds.internic.net/rfc/rfc1156.txt.

[80] M. J. McTiffin, A. P. Hulbert, T. J. Ketseoglou, W. Heimsch, and G. Crisp. Mobile Access to an ATM Network Using a CDMA Air Interface. *IEEE Journal of Selected Areas in Communication*, June 1994.

[81] B. Meandzija and J. Westcott, editors. *Guidelines for Structuring Manageable Entities*. North-Holland, 1989.

[82] D. L. Mills, editor. *Network Time Protocol*. M/A-COM Linkabit, 1985. Online version available at http://ds.internic.net/rfc/rfc958.txt.

[83] Mahmoud Naghshineh. The Virtual Connection Tree - A Scheme for Routing, Resource Allocation, Call Admission, and QoS Provisioning in Mobile/Wireless ATM Networks. In *Proceedings of Wireless ATM Networking Workshop*, June 1996.

[84] B. L. Noble and R. D. Chamberlain. Predicting the Future: Resource Requirements and Predictive Optimism. In *Proceedings of the 9th Workshop on Parallel and Distributed Simulation*, pages 157,164, 1995.

[85] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw Hill, 1991.

[86] C. Perkins, editor. *IP Mobility Support*. Mobile-IP Working Group, October 1996. Online version available at http://ds.internic.net/rfc/rfc2002.txt.

[87] James L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, 1981.

[88] S. K. Biswas R. Yuan and D. Raychaudhuri. A Signaling and Control Architecture for Mobility Support in Wireless ATM Networks. In *Proceeding of ICC'96*, pages 478,484, June 1996.

[89] H. Rajaei, R. Ayani, and L. E. Thorelli. The Local Time Warp Approach to Parallel Simulation. In *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, pages 37–43, 1993.

[90] Hassan Rajaei, Rasul Ayani, and Lars-Erik Thorelli. The Local Time Warp Approach to Parallel Simulation. In *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, pages 119,126, 1993.

[91] Dipankar Raychaudhuri and Newman D. Wilson. ATM-Based Transport Architecture for Multi-services Wireless Personal Communication Networks. *IEEE Journal of Selected Areas in Communication*, June 1994.

[92] Wolfgang Reisig. *Petri Nets*. Springer-Verlag, 1985.

[93] Thomas L. Rodeheffer and Michael D. Schroeder. Automatic Reconfiguration in Autonet. In *Proceedings of the 13th ACM Symposium on Operating System Principles*, pages 183–187, 1991.

[94] Marshall T. Rose. *The Simple Book, An Introduction to the Management of TCP/IP Based Internets*. Prentice Hall, 1991.

[95] Routing over Large Clouds Working Group. *NBMA Next Hop Resolution Protocol (NHRP)*, 1995. Online version available at gopher://ds.internic.net/00/-internet-drafts/draft-ietf-rolc-nhrp-04.txt.

[96] F. Santucci and N. Benvenuto. A Least Squares Path Loss Estimation Approach to Handover Algorithms. In *Proceeding of ICC'96*, pages 802,806, June 1996.

[97] B. Kean Sawhill. Self-Organizing Criticality and Complexity Theory. In Lynn Nadel and Daniul Stien, editors, *1993 Lectures in Complex Systems*, pages 143,170. Addison-Wesley, 1995.

[98] Michael D. Schroeder, Andrew D. Birrell, Michael Burrows, Hal Murray, Roger M. Needham, Thomas L. Rodeheffer, Ed Satterthwaite, and Chuck Thacker. Autonet: A High-Speed, Self-Configuring Local Area Network Using Point-to-Point Links. *IEEE Selected Areas in Communications*, pages 183–187, October 1991.

[99] Srinivasan Seshan, Hari Balakrishnan, and Randy H. Katz. Handoffs in Cellular Wireless Networks: The Daedalus Implementation Experience. *Kluwer International Journal on Wireless Personal Communications*, 1996.

[100] Joel Short, Rajive Bagrodia, and Leonard Kleinrock. Mobile Wireless Network System Simulation. In *Mobicom '95*, 1995.

[101] L. M. Sokol, D. P. Briscoe, and A. P. Wieland. MTW: A Strategy for Scheduling Discrete Simulation Events for Concurrent Execution. In *Proceedings of the 2nd Workshop on Parallel and Distributed Simulation*, pages 34–42, 1988.

[102] L. M. Sokol and B. K. Stucky. WOLF: A Rollback Algorithm for Optimistic Distributed Simulation Systems. In *Proc. 1990 SCS Multi conference on Distributed Simulation*, pages 169–173, 1990.

[103] Sudhir Srinivisan and Jr. Paul F. Reynolds. Adaptive Algorithms vs. Time Warp: An Analytical Comparison. Technical Report CS-95-20, University of Virginia, April 1995.

[104] Sudhir Srinivisan and Jr. Paul F. Reynolds. NPSI Adaptive Synchronization Algorithms for PDES. Technical Report CS-94-44, University of Virginia, April 1995.

[105] L. Steinber, editor. *Techniques for Managing Asynchronously Generated Alerts*. IETF, May 1991.

[106] J. S. Steinman. SPEEDES: A Unified Approach to Parallel Simulation. In *Proceedings of the 6th Workshop on Parallel and Distributed Simulation*, pages 75,84, 1992.

[107] J. S. Steinman. Breathing Time Warp. In *Proceedings of the 7th Workshop on Parallel and Distributed Simulation*, pages 109,118, 1993.

[108] Hiroshi Suzuki, editor. *atm96-0530.txt*. ATM Forum, 1996.

[109] H. Takagi. *An Analysis of Polling Systems*. MIT Press, 1986.

[110] David L. Tennenhouse and Vanu G. Bose. SpectrumWare: A Software-Oriented Approach to Wireless Signal Processing. In *Mobicom '95*, 1995.

[111] David L. Tennenhouse and David J. Wetherall. Towards an Active Network Architecture, 1996.

[112] R. Thomas, H. Gilbert, and G. Mazziotto. Influence of the Movement of Mobile Stations on the Performance of the Radio Cellular Network. *Proceedings of the 3rd Nordic Seminar,* September 1988.

[113] Pete Tinker and Jonathan Agra. Adaptive Model Prediction Using Time Warp. In *SCS '90*, 1990.

[114] John Turnbull. A Performance Study of Jefferson's Time Warp Scheme for Distributed Simulation. Master's thesis, Case Western Reserve University, May 1992.

[115] Kimmo Varpaniemi, Jaakko Halme, Kari Hiekkanen, and Tino Pyssysalo. PROD Reference Manual. Technical Report ISBN 951-22-27070-X, Helsinki University of Technology, August 1995.

[116] M. Veerarghavan, M. J. Karol, and K. Y. Eng. Mobility and Connection Management in a Wireless ATM LAN. *IEEE Journal of Selected Areas in Communication*, 15(1), January 1997.

[117] K. Voss, H. J. Genrich, and G. Rozenberg. *Concurrency and Nets*. Springer-Verlag, 1987. ISBN 0-387-18057-5.

[118] M. Mitchell Waldrop. *Complexity: The Emerging Science at the Edge of Order and Chaos*. Simon & Schuster, 1992. ISBN 0-671-76789-5.

[119] Shyhtsun F. Wu and Gail E. Kaiser. Network Management with Consistently Managed Objects. Technical Report CUCS-013-90, Columbia University, 1990.

[120] Shyhtsun F. Wu and Gail E. Kaiser. On Hard Real-Time Management Information. Technical Report CUCS-013-90, Columbia University, 1990.